INTRUSION DETECTION: EMBEDDED SOFTWARE MACHINE LEARNING AND HARDWARE RULES BASED CO-DESIGNS

Razan Abdulhammed

Under the Supervision of

Dr. Miad Faezipour

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIRMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOHPY IN COMPUTER SCIENCE

AND ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

May, 2019

INTRUSION DETECTION: EMBEDDED SOFTWARE MACHINE LEARNING AND HARDWARE RULES BASED CO-DESIGNS

Razan Abdulhammed

Under the Supervision of Dr. Miad Faezipour

Approvals

Committee Members

Name	**Signature**	**Date**
Dr. Miad Faezipour	Miad Faezipour	4-23-2019
Dr. Abdel-shakour Abuzneid	Abdelshakour Abuzneid	4-23-2019
Dr. Ausif Mahmood	Ausif Mahmood	4-23-2019
Dr. Xingguo Xiong	Xingguo Xiong	04/23/2019
Dr. Amir Esmailpour	Amir Esmailpour	04/23/2019
Ph.D. Program Coordinator		
Dr. Khaled M. Elleithy	Khaled Elleithy	5/17/19
Chairman, Computer Science and Engineering Department		
Dr. Ausif Mahmood	Ausif Mahmood	4-23-2019
Dean, School of Engineering		
Dr. Tarek M. Sobh	T. Sobh	5/17/19

INTRUSION DETECTION: EMBEDDED SOFTWARE MACHINE LEARNING AND HARDWARE RULES BASED CO-DESIGNS

© Copyright by Razan Abdulhammed 2019

INTRUSION DETECTION: EMBEDDED SOFTWARE MACHINE LEARNING AND HARDWARE RULES BASED CO-DESIGNS

ABSTRACT

Security of innovative technologies in future generation networks such as (Cyber Physical Systems (CPS) and Wi-Fi has become a critical universal issue for individuals, economy, enterprises, organizations and governments. The rate of cyber-attacks has increased dramatically, and the tactics used by the attackers are continuing to evolve and have become ingenious during the attacks. Intrusion Detection is one of the solutions against these attacks. One approach in designing an intrusion detection system (IDS) is software-based machine learning. Such approach can predict and detect threats before they result in major security incidents. Moreover, despite the considerable research in machine learning based designs, there is still a relatively small body of literature that is concerned with imbalanced class distributions from the intrusion detection system perspective. In addition, it is necessary to have an effective performance metric that can compare multiple multi-class as well as binary-class systems with respect to class distribution. Furthermore, the expectant detection techniques must have the ability to identify real attacks from random defects, ingrained defects in the design, misconfigurations of the system devices, system faults, human errors, and software implementation errors. Moreover, a lightweight IDS that is small, real-time, flexible and reconfigurable enough to be used as permanent elements of the system's security infrastructure is essential.

The main goal of the current study is to design an effective and accurate in-

trusion detection framework with minimum features that are more discriminative and representative. Three publicly available datasets representing variant networking environments are adopted which also reflect realistic imbalanced class distributions as well as updated attack patterns. The presented intrusion detection framework is composed of three main modules: feature selection and dimensionality reduction, handling imbalanced class distributions, and classification. The feature selection mechanism utilizes searching algorithms and correlation based subset evaluation techniques, whereas the feature dimensionality reduction part utilizes principal component analysis and auto-encoder as an instance of deep learning. Various classifiers, including eight single-learning classifiers, four ensemble classifiers, one stacked classifier, and five imbalanced class handling approaches are evaluated to identify the most efficient and accurate one(s) for the proposed intrusion detection framework.

A hardware-based approach to detect malicious behaviors of sensors and actuators embedded in medical devices, in which the safety of the patient is critical and of utmost importance, is additionally proposed. The idea is based on a methodology that transforms a device's behavior rules into a state machine to build a Behavior Specification Rules Monitoring (BSRM) tool for four medical devices. Simulation and synthesis results demonstrate that the BSRM tool can effectively identify the expected normal behavior of the device and detect any deviation from its normal behavior. The performance of the BSRM approach has also been compared with a machine learning based approach for the same problem. The FPGA module of the BSRM can be embedded in medical devices as an IDS and can be further integrated with the machine learning based approach. The reconfigurable nature of the FPGA chip adds an extra advantage to the designed model in which the behavior rules can be easily updated and tailored according to the requirements of the device, patient, treatment algorithm, and/or pervasive healthcare application.

Dedicated to the memory of my wonderful deeply missed sister (Eftekhar)

Forever you remain in my soul

ACKNOWLEDGEMENTS

First and foremost, praises and thanks to the God, the Almighty, for his blessings throughout my dissertation work to complete my studies successfully. This dissertation comes to its conclusion due to the assistance, guidance and trust of several people. I would like to thank all of them. First, I would like to express my heartfelt gratitude to my supervisor Dr. Miad Faezipour, who was not only my mentor, but a friend as well. Her valuable advice, guidance, scientific and moral support during my research has not been only inspirational but also determinant in achieving my goals. I could not be prouder of my academic roots and hope that I can in turn pass on the research values and the dreams that she has given to me.

I would like to offer my deep appreciation to Dr. Abdel-shakour Abuzneid, Dr. Ausif Mahmood, Dr. Xingguo Xiong, and Dr. Amir Esmailpour for serving on my supervisory committee, taking the time to evaluate this dissertation, and providing their valuable feedback and comments.

I am extremely grateful to my sister and my brothers for their love, prayers, caring and continuing support such that I am able to complete this dissertation work.

I would like to say thanks to my friend Najwa Alahmadi, her husbend Waleed alharbi, and charming children Hala, Qamar, and Ahmed.

Last but not the least, I would like to thank my husband (Abdulhakeem) and my beloved son (Abdulaziz) for their empathy, great sense of humor, and supporting me spiritually throughout writing this thesis and my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ABD Anomaly Based Detection

Acc Accuracy

AE Auto-Encoder

$(AE - BN)_{Bc}$ Auto-Encoder-Bayesian Network-Binary-Class

$(AE - BN)_{Mc}$ Auto-Encoder-Bayesian Network-Multi-Class

$(AE - LDA)_{Bc}$ Auto-Encoder-Linear Discriminant Analysis-Binary-Class

$(AE - LDA)_{Mc}$ Auto-Encoder-Linear Discriminant Analysis-Multi-Class

$(AE - QDA)_{Bc}$ Auto-Encoder-Quadratic Discriminant Analysis-Binary-Class

$(AE - QDA)_{Mc}$ Auto-Encoder-Quadratic Discriminant Analysis-Multi-Class

$(AE - RF)_{Bc}$ Auto-Encoder-Random Forest-Binary-Class

$(AE - RF)_{Mc}$ Auto-Encoder-Random Forest-Multi-Class

AGS Attribute Group Set

ARP Address Resolution Protocol

ANN Artificial Neural Network

BFS Best First Search

BFE Backward Feature Elimination

BSRM Behavior Specification Rules Monitoring

CB Class Balancer

CD Cardiac Device

CFS Correlation Feature Selection

CFsSubsetEval Correlation Feature Subset Evaluate

CICIDS2017 Canadian Institute for Cybersecurity Intrusion Detection System 2017

CIDDS-001 Coburg Intrusion Detection System dataset-001

CGM Continous Glocuse Monitor

$CM_{(Bc)}$ Combined Metric for Binary Class

$CM_{(Mc)}$ Combined Metric for MultiClass

CNN Convolutional Neural Network

CPS Cyber Physical System

CRI Classification Rule Induction

CTree Conditional inference trees

DBN Deep Belief Network

DDoS Distributed Denial of Service

DNN Deep Neural Network

DR Detection Rate

DoS Denial of Service

DT Decision Tree

FAR False Alarm Rate

FFC Forward Feature Construct

F-M F-Measure

FPGA Field Programmable Gate Array

FPR False Positive Rate

FS Feature Set

FSG Feature Set Group

FTP File Transfer Protocol

FSM Finite State Machine

GA Genetic Algorthim

G-M Geometric-Mean

HBD Hybrid Based Detection

HS Harmony Search

HTTP Hyper Text Transfer Protocol

HTTPS Secure Hyper Text Transfer Protocol

ID3 Iterative Dichotomiser 3

IDS Intrusion Detection System

IoT Internet of Things

IV Initialization Vector

KNN K-Nearest Neighbor

LDA Linear Discriminant Analysis

LSTM Long Short Term Memory

MCC Matthews Correlation Coefficient

M-i-M Man in the Middle

ML Machine Learning

MLP Multi Layer Perceptron

MVR Missing Value Ratio

PCA Principal Component Analysis

$(PCA-BN)_{Bc}$ Principal Component Analysis-Bayesian Network-Binary-Class

$(PCA-BN)_{Mc}$ Principal Component Analysis-Bayesian Network-Multi-Class

$(PCA-LDA)_{Bc}$ Principal Component Analysis-Linear Discriminant Analysis-Binary-Class

$(PCA-LDA)_{Mc}$ Principal Component Analysis-Linear Discriminant Analysis-Multi-Class

$(PCA-RF)_{Bc}$ Principal Component Analysis-Random Forest-Binary-Class

$(PCA-RF)_{Mc}$ Principal Component Analysis-Random Forest-Multi-Class

$(PCA-QDA)_{Bc}$ Principal Component Analysis-Quadratic Discriminant Analysis-Binary-Class

$(PCA-QDA)_{Mc}$ Principal Component Analysis-Quadratic Discriminant Analysis-Multi-Class

PCAg Patient Control Analgesia

PCAP Packet CAPTure

PRC Area Under Precision Recall Curve

PVS Prototype Verification System

QDA Quadratic Discriminant Analysis

RF Random Forest

ROC Area Under Receiver Operating Characteristic Curve

RR Reduction Ratio

RMSE Root Mean Squared Error ***RWR*** Re-sample With Replacement

RWoR Re-sample Without Replacement

SAE Sparse Auto-Encoder

SBD Signature Based Detection

SBD Specification Based Detection

SIP Swarm Intelligence Optimization

SMOTE Synthetic Minority Oversampling Technique

SSH Secure Socket Shell ***SPA*** Statistical Power Analysis

SVM Support Vector Machine

TPR True Positive Rate

TICS Traditional Information Communication System

UDBB Uniform Distribution Based Balancing

VSM Vital Signs Monitor

WHICD With Handling Imbalanced Class Distribution

WoHICD Without Handling Imbalanced Class Distribution

WSRT Wilcoxon Signed Ranks Test

XGBoost eXtreme Gradient Boosting

CHAPTER 1: INTRODUCTION

An Intrusion Detection System (IDS) is a software-based application or a hardware device that is used to identify malicious behavior (which is commonly known as attacks) in the network. Through years, the types of attacks have increased dramatically in the network, and the tactics used by the attackers are continuing to evolve and become ingenious during the attacks. Thus, IDS is a necessary addition to the security infrastructure, and the IDS must accommodate new diversified types of attacks, security threats, and new vulnerabilities as well as handle older attacks that have not disappeared or have not evolved. IDS allows the organizations and/or networks to protect their systems and/or devices. For decades, IDS developers employed various approaches to build an IDS. One of these approaches is Machine Learning (ML) [1], and more specifically; classification to detect cyber-attacks. This approach requires a dataset that encompasses training and testing data. In this contest, this dissertation reviewed the existing datasets and its usage in IDS and ML between 2014 and 2018. The review includes six datasets, namely: AWID [2], UNSW-NB15 [3, 4, 5], GPRS [6], CIDDS-001 [7] , CICIDS2017 [8] and UGR'16 [9], as well as 55 research articles. Descriptive statistics about the reviewed studies are stated which include: the applied algorithms, the size and type of the used dataset for training and testing, classification output classes as binary or multi-class, and whether the dataset has balanced or imbalanced class distributions as tabulated in Table 1.1.

A great deal of previous research regarding IDS has focused on binary classifi-

Table 1.1: Descriptive information related to reviewed dataset

Dataset	Labeling	Type	Missing Value	Dataset Split	Format	Skew ratio	Environment	Features
AWID	Labeled	Multi-class	Yes	Train and Test Files	Packet	3:1	Wi-Fi	154
CICIDS2017	Labeled	Multi-class	No	Single File	Packet, Bi- directional Flow	5:1	TICS	83
CIDDS-001	Labeled	Binary-class	No	Train and Test Files	Uni- directional Flow	10:1	TICS	14

cation. Moreover, previous studies of intrusion detection systems have not dealt with classification of network traffic data with imbalanced class distribution. The issues pertaining to imbalanced classes and frequent occurrence of such cases call for further research efforts. This dissertation considers multi-class datasets and imbalanced class distributions. Imbalanced data can be numerically defined by the skew ratio between the classes [10].

Machine learning, and more specifically, classification of network traffic, is an effective approach to automate and simplify the development of intrusion detection. Recently, there has been a dramatic increase in security threats and attacks. Furthermore, the imbalanced class distribution problem is prevalent in network traffic records. These records have different types of attacks; some of these are present; some overwhelming; whereas, other attacks are somehow rare in the collection of network connection traffic records. For instance, the AWID-ATK-R dataset [2] contains fifteen types of attacks: Among these fifteen types of attacks, the power-saving, CTS, RTS, and Chop-chop are intrinsically rare in their class distribution compared to other types of attacks.

In the literature, Intrusions detection performance can be evaluated through a set of metrics. There are many well known performance measures such as False Positive Rate (FPR), True Positive Rate (TPR,) F-Measure [11], Matthews Correlation Coefficient (MCC) [12], Cohen Kappa [13, 14, 15, 16], Area Under Receiver Operating Characteristic Curve (ROC) [13, 16], Area Under Precision Recall Curve (PRC), and Accuracy.

Usually, the overall accuracy is used to measure the effectiveness of a classifier.

Unfortunately, because of the presence of imbalanced data, the overall accuracy may fail to provide adequate information about the performance of the classifier. Furthermore, the accuracy is very sensitive to the class distribution and might be misleading in some way. This becomes even more complicated within multi-class problems. In this dissertation, we evaluate the performance in multi-class and binary-class schemes. Since the accuracy is not an adequate measure of the effectiveness of the classifier model, we calculate the geometric mean [17, 18, 19] (G-mean) as the higher root of the product of sensitivity for each class. Furthermore, this dissertation extends the combined metric [20] such that it can work for multi-class problems.

1.1 Research Problem and Scope

This dissertation aims to address research challenges of designing efficient intrusion detection systems, mainly based on machine learning approaches. Our goal is to also design and validate intrusion detection utilization in current and imbalanced class distribution datasets [21]. Furthermore, as an IDS case study, it examines behavior specification rule-based detection [22], and investigates the impact of using a hardware approach to detect malicious behavior in sensors and actuators that are embedded in medical devices.

1.2 Motivation and Objective

Network Security has become a critical universal issue for individuals, economy, enterprises, organizations and governments. The rate of cyber-attacks has increased dramatically, and the tactics used by the attackers are continuing to evolve and have become ingenious during attacks. Intrusion Detection Systems (IDS) are one of the solutions against these attacks. Furthermore, innovative technologies of future

generation networks such as Cyber Physical System (CPS), and Wireless network commonly known as Wi-Fi have emerged, which require a distinguished understanding of the main challenges and constraints that face the design and implementation of an IDS for such systems. Thus, IDS always need to improve its performance in terms of increasing the accuracy and decreasing false alarms. In machine learning based IDS, integrating efficient feature selection and feature dimensionality reduction with intrusion detection has shown to be a successful approach since it can help in selecting the most informative features and/or reduce the feature dimensionality from the complete set of features. More specifically, classification of network traffic data with imbalanced class distribution has posed a significant drawback of the performance attainable by most well-known classifier learning algorithms, which assume a relatively balanced class distribution and equal misclassification costs. Imbalanced class distribution and frequent occurrences of such cases call for more research and investigation. The objective of this study is to utilize various techniques to build an IDS with the aim to advance the classification of imbalanced data.

1.3 Contributions

Despite of the amount of research that has been done on intrusion detection systems to improve the concerns associated with the Accuracy, Recall, Precision, and False Negative Rates, it is still a challenging task which requires more investigation. Moreover, relatively little research has been carried out on multi-class classification, and even less on imbalanced datasets. The goal of this dissertation is to propose workable solutions to this problem by offering new methodologies for multi-class and binary-class classification. The key contributions of this dissertation include:

(1) It provides a review of the existing datasets for ML-based IDS between 2014 and 2018. The aim of providing this review is to identify the available datasets

that can be utilized to build an IDS for different networking systems. In addition, guidelines are presented for selecting the appropriate dataset that can serve as a general reference for IDS researchers.

(2) It develops a new performance metric ($Combined_{Mc}$) to evaluate the performance of intrusion detection systems. The proposed metric will enable researchers to compare multi-class systems by incorporating individual class distributions and three metrics which are accuracy, detection rate and false alarm rate.

(3) It extracts four different number of selected Features Sets (FS): 32-FS, 10-FS, 7-FS, and 5-FS. The key contributions among these feature sets include extracting a new subset of 5 features that produce high accuracy with minimum false positives (FP) and validated using the 10-fold cross validation approach.

(4) In addition, a thorough investigation of the extracted features set performance using the most common machine-learning based classifiers used in the literature, namely AdaBoost [23, 24], Random Forest [25], Random Tree [16], J48 [26, 27], Logit Boost [28, 29, 30], Multi-Layer Perceptron [31, 32], ZeroR, OneR, and Simple logistic is carefully studied and experimentally evaluated.

(5) It reduces the CICIDS2017 dataset's feature dimensions from 81 to 10, while maintaining a high accuracy of 99.6% in multi-class and binary classification.

(6) It highlights the importance of imbalanced class problems in network traffic for devising an effective IDS, and studies the effect of various sampling techniques on the CIDDS-001 dataset along with different classification approaches. Moreover, it explores the influence of class distribution on other IDS datasets as well, and utilizes sampling techniques for balancing the datasets in multi-class and binary classes.

(7) A uniform distribution based balancing (UDBB) approach is developed to handle the imbalanced distribution of the minority class instances in the CICIDS2017 network intrusion dataset.

(8) As a real-life IDS application and a case study, this work also proposes and analyzes a hardware-based specification rule approach using an FPGA module for malicious behavior of sensors and actuators embedded in medical devices to protect the safety of patients. The performance of machine learning based approaches have also been tested for the same application. The FPGA module can be embedded within medical devices so that the device that is being monitored for its behavior, can easily be seen for any deviation from its behavior specification. This is critical and of utmost importance for patients' safety.

1.4 Dissertation Outline

The remainder of this dissertation is structured as follows. Chapter 2 introduces the background, taxonomy and literature survey related to machine learning based intrusion detection as well as the relevant existing datasets and an introduction to relevant cyber physical systems. Chapter 3 is concerned with the methodology used for this study. Chapter 4 explores the implementation and test plan. Next, the results and findings of the research are presented and discussed in Chapter 5. Afterwards, statistical analysis is performed and discussed in Chapter 6. Finally, Chapter 7 provides the conclusions, brief summary, and critiques of the research findings. Areas for further research are also identified in the same chapter.

CHAPTER 2: BACKGROUND, TAXONOMY AND LITERATURE SURVEY

This chapter begins by providing a background overview and taxonomy of Intrusion Detection Systems (IDS). Then, it examines imbalanced class problems in IDS's. Next, it goes on to a review of existing datasets in machine learning based IDS's between 2014 and 2018. Afterwards, it attempts to provide a brief summary of the structure, attacks, as well as the state-of-the-art work related to the AWID dataset [2]. Then, the overall structure, attacks, and relevant work of the CICIDS2017 dataset is provided. Following CICIDS2017, a brief overview of the recent CIDDS-001 related work is also provided. Then, previous work investigating imbalanced classes in IDS's have been studied. Finally, a summary relevant to a sample of IDS for Cyber Physical Systems is provided.

2.1 Intrusion Detection System (IDS)

Intrusion Detection System (IDS) can be defined as the process of detecting and identifying the non-permitted access to the services, activities, system information and resources of network systems [33]. This concept was first described in the early eighties [34]. In general, and from a system architecture perspective, the core functions of an intrusion detection system consists of three essential elements [35], as shown in Figure 2.1; Collecting data concerning adversary, Analyzing the data, and

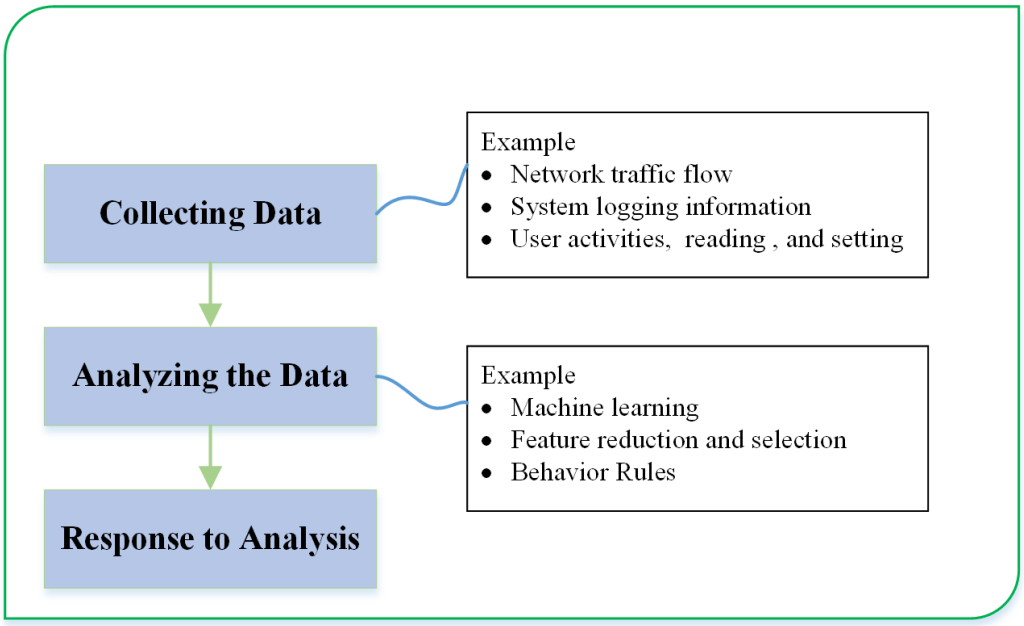

Figure 2.1: IDS core functions

Responding to the analysis. Examples of adversary data collection may include system logging information from a local host or multi-hosts, user activity on operating systems and the recording traffic received and sent on network interfaces, or it can be of both types. Analysis may include statistical methods such as Markova Process Marker, Multivariate, Unvigilant, Time series and Operational Statistical Moment [35]. Another example of analysis is machine learning which includes techniques such as bayesian network, genetic algorithm, neural network, fuzzy logic and outlier detection. Other examples of analysis are string and pattern matching, probabilistic analysis, and data mining which include frequent pattern mining, classification, clustering, and mining data streams. Examples of responses may include spreading an alarm, updating routing tables, and closing a session. The first two core functions have been treated numerously in the literature. In contrast, the third function was less treated [35, 36].

2.2 IDS Classification Tree Hierarchy

Moving up in this section, a hierarchical classification tree is developed to organize the sample of the existing IDS's, and are chosen carefully to reflect the extensive groups of research work under investigation. Also, these classification criteria are important as it represents a solid scientific base that group and bind certain IDS's together. Figure 2.2 shows the hierarchical classification tree of IDS based on three classification criteria [37]:

(1) Time-line: A measure that identifies "When" the analyzing process takes place to detect the intrusions.

(2) Audit Material: A criterion that determines "How" the information collection process is accomplished for data analysis.

(3) Detection approach: A standard that identifies "How" the malicious activities will be interfered by the IDS for identifying the intrusions.

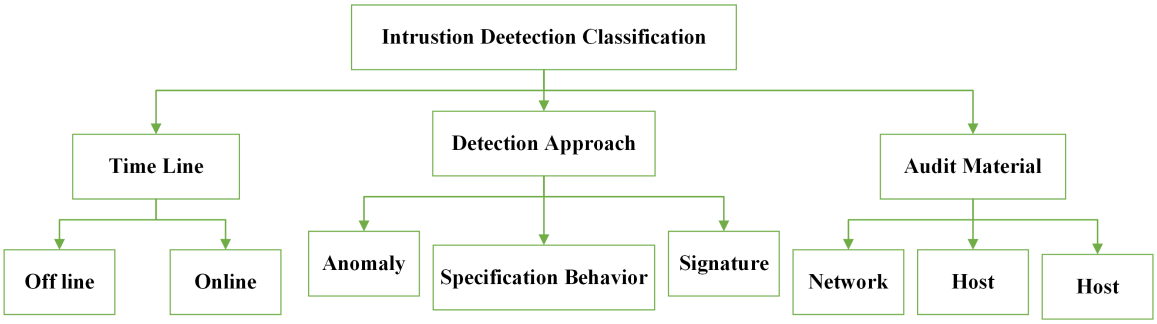

Figure 2.2: IDS classification tree

2.2.1 Time Line

According to the Time-line criterion, IDS can be grouped into real-time IDS and offline IDS. In a real-time approach, the analyzing process takes place while the

sessions are ongoing, and the IDS immediately sets on its alarm to indicate that an attack is detected. In an offline IDS, the analyzing process takes place after the information has been already collected.

2.2.2 Audit Material

The analyzed data (audit material) can be collected by different schemes [37]. These include: host-based, network-based, and hybrid-based approaches.

2.2.3 Detection Approach

In general, IDS's can be grouped based on the detection approach into: signature-based detection, anomaly-based detection, and specification behavior-based detection. Signature-based detection maintains a database of various known attack signatures. Thus, it reveals the use of patterns from prior known threats. On the other hand, anomaly-based detection methods examine and recognize intrusion behavior based on verifying the prior activities and checks for any deviations from the normal traffic behavior [38]. Detection approaches are specification behavior-based, hybrid-based, signature-based, anomaly-based, and cross-layer based approaches. These approaches have their own pros and cons which make them suitable for a specific type of networking environment. Next, detailed discussion about each classification approach is given.

2.2.3.1 Anomaly-Based Detection (ABD)

This technique was proposed in 1987 [39]. The fundamental concept behind this technique is to define the behavior of the network and/or system, and then this predefined behavior is compared with the normal behavior. The result will be either to accept it or it will trigger the alarm management system for further investigation.

The function of the anomaly-based IDS is performed by two phases, which are (1) training phase and (2) detection phase. A normal profile of the network traffic (network behavior) and/or system information logs are generated throughout the training phase. In the detection phase, the actual traffic is matched to the current normal profile that searches for any deviations. The network administrator and the network security experts prepare the accepted network behavior profiles. The constructed profiles are based on users' logging information, servers' logging information, and network connection features such as protocol type, flags and so on. The applicability of this approach is defined by the attribute's nature as well as the features of the targeted system under investigation. Based on processing methods [40], the Anomaly Based Detection (ABD) can be categorized into (1) data mining-based [41, 42], (2) statistical-based, (3) traffic analysis-based systems, (4) probabilistic-based, and (5) machine learning based. The dissertation does not engage with statistical-based, traffic analysis-based systems, and probabilistic-based. methods. This dissertation will focus on machine learning based approaches and hardware rules based co-designs.

2.2.3.2 Signature-Based Detection (SBD)

This technique, which is a pattern-based detection [36] approach, implements an intruder profiling mechanism that looks for run time features that correspond to a pre-determined unique feature of misbehavior actions or attacks. These techniques are recognized not only by their low false positive rates, but also by their ability to detect non-zero day attacks with high accuracy. On the contrary, it cannot pinpoint zero-day attacks or adjusted attacks. The main research challenge in signature-based detection is not only to create a powerful attack dictionary, but also to add new attack patterns. Compared to the anomaly-based detection technique, this technique requires more computation and resources [43].

2.2.3.3 Hybrid-Based Detection (HBD)

In 2007, a study by Kai et al. [44] proposed the design principles and evaluation results of the Hybrid Intrusion Detection System (HIDS). The basic concept in this method is to use a combination of two or more previously mentioned methods. Many researchers use this technique to implement an IDS model which helps in enhancing the system's detection ability to disclose novel as well as known security attacks [45].

2.2.3.4 Specification Behavior Detection (SBD)

In 2003, specification based detection methodology was introduced by [46, 47], which provided the capability to detect non-zero day attacks as well as zero-day attacks, while exhibiting a low false positive rate. In this method, the detection process is accomplished through developing the behavioral specification of legitimate system behaviors manually. This approach is used as a basis for both detecting attacks and characterizing the legitimate system behavior. The intruder activity can be identified by observing the normal system and/or users' behavior of the targeted system under investigation. It can recognize endeavors to exploit new and unexpected vulnerabilities as well as recognizes misuse of privileges kinds of attacks which do not really take advantage of any security vulnerability [47]. One of the advantages of this method is that it can avoid false positives since the specification can capture all legitimate behavior. However, developing an error-free, complete and detailed specification for the system is a challenging task. The approaches that are used with this method can be statistics, neural networks, expert systems, computer immunology, state machines or extended finite state automata, and user intention.

2.3 Imbalanced Class Problems in IDS

The problem of learning from skewed multi-class datasets is an important topic that arises very often in practice in classification problems. In such problems, almost all the instances are labeled as one class (called the majority, or negative class), while far fewer instances are labeled as the other class or classes (often called the minority class(es), or positive class(es)); usually the more important class(es).

2.4 Review of Existing Datasets Usage in IDS and ML

New cases of intrusions, new bugs, security issues and vulnerabilities are evolving every day. Consequently, developers in the domains of intrusion detection are constantly designing new methods to minimize the effect of these security issues. However, accessing suitable datasets for evaluating various research designs in these domains is a major challenge for the research community, vendors and data donors over the years [8]. According to Nehinbe [48], these challenges are data privacy issues, getting approval from the data owners, scope of evaluative datasets, different research objectives, problem of documentation, understanding the datasets, data labeling, availability of evaluative datasets, and finally discrepancies in evaluative datasets. Moreover, in order to design an efficient machine learning based IDS, it is important to build the classification models through representative datasets [49].

To sum up, there are numerous datasets available in the literature. Some of these datasets are not available for public access such as IRSC [51] and SANTA[52]. Others such as KDDCUP99 [53, 54, 55], which was released in 1999, and NSL-KDD [56] that was released in 2009 have grown increasingly irrelevant as new attack types and methods have emerged that are not reflected by these older datasets.

Considering all the points mentioned, this dissertation presents the sample of

Table 2.1: Sample of existing IDS dataset citations from 2014 to 2018

Year	Dataset	Environment	Citations Count
2015	UNSW-NB15 [5, 50]	TICS (Traditional Information Communication Systems)	36
2015	AWID [2]	Wi-Fi	18
2014	GPRS [6]	Wi-Fi	4
2016	CIDDS-001 [7]	TICS	4
2016	UGR'16 [9]	TICS	1
2017	CICIDS2017 [8]	TICS	10

the existing datasets in Table 2.1. Moreover, it reviews the usage of these datasets in ML-based IDS and the findings are summarized in Tables 2.2 and 2.3.

2.5 AWID-ATK-R Structure

AWID-ATK-R, a subset of AWID, is a labeled dataset with a total number of 155 features. AWID-ATK-R was collected based on real traces of normal and intrusion activities of the 802.11 Wi-Fi network [2] and it has a finer grained class labeling corresponding to the attack name. The training set consists of 10 classes namely Amok, Arp, Authentication request, Beacon, Caffe latte, Deauthentication, Evil twin, Frag- mentation, Probe response and Normal. The total number of records in the training set is 1,765,000. The normal traffic encompasses 1,633,190 records while the attacks records are 162,358. On the other hand, the total number of records in the test set is 575,643 records. The normal traffic comprises of 530,785 records. At the same time, the attacks records are 44,585. In addition, the test set has 15 classes with 7 different classes compared to the training set, namely Chop- chop, CTS, Disassociation, Hirte, Power-saving, Probe request and RTS. Furthermore, Probe response and Authentication- request are included in the training set only. Table 3 highlights the characteristics of the AWID-ATK-R subset.

Table 2.2: Review of existing datasets' usage in machine learning based IDS between 2014 and 2018

Approaches	Dataset	Year
SVM (Classification)+ GA (Feature selection) [57]	UNSW-NB15	2016
ANN+GA [58]	UNSW-NB15	2016
Multi-class Cascade ANN [59]	UNSW-NB15	2017
ANN [60]	UNSW-NB15	2017
Multiscale Hebbian neural network [61]	UNSW-NB15	2017
Geometric Area Analysis and Big Data [62]	UNSW-NB15	2017
Feature Selection + Central points [63]	UNSW-NB15	2017
K-Support Vector Classification-Regression [64]	UNSW-NB15	2017
Decision engine [65]	UNSW-NB15	2017
Features Selection [66]	UNSW-NB15	2017
Beta Mixture Models and Outlier Detection [67]	UNSW-NB15	2016
Repeated Sampling and Clustering +Cross Validation [68]	UNSW-NB15	2016
Bagged trees, RUSBoost, LogitBoost, GentleBoost [69]	UNSW-NB15	2017
Cognitive Approach, Classification [70]	UNSW-NB15+AWID	2017
ANN [71]	UNSW-NB15	2017
Binary Tree+SVM [72]	UNSW-NB15	2016
ANN+Naïve Bayes, C.45 Association Rule[73]	UNSW-NB15	2017
Multiverse optimization+ANN [74]	UNSW-NB15	2017
SVM+KNN+Decision Trees+Rapper [75]	UNSW-NB15	2017
Deep Learning [76]	UNSW-NB15	2017
LogicitBoost [29]	UNSW-NB15	2017
REpTree [30]	UNSW-NB15	2017
Honeypots+Fuzzy hashing [31]	UNSW-NB15	2017
Gaussian Mixture Model (GMM) [77]	UNSW-NB15	2017
Ontology-based+Intuitionistic Fuzzy Logic (IFL) [78]	UNSW-NB15	2016
MLP [79]	UNSW-NB15	2017
GA+trees driven rule induction [80]	UNSW-NB15	2018
Software Defined Networking [81]	UNSW-NB15	2017
GA+logistic regression+decision tree [82]	UNSW-NB15	2017
logistic Regression+SVM-RBF+ SVM-Polynomial [83]	UNSW-NB15	2016
Security architecture [84]	UNSW-NB15	2016
k-means algorithm [85]	UNSW-NB15	2017
Linear regression (LR) + random forest (RF) [86]	UNSW-NB15	2017
(k-NN)+Gradient Descent ANN+Hebbian algorithm [87]	UNSW-NB15	2017
J48, AdaBoost, OneR [2]	AWID-CLS-R	2015
Deep Abstraction+SAE+Weighted feature selection [88]	AWID-CLS-R	2018

2.5.1 Attacks in AWID

In this dissertation, AWID was selected as one of the benchmark datasets for its reliability, validity, and it's highly imbalanced class distribution. To the best of our knowledge, this dataset is considered as a unique dataset tied to the Wi-Fi

Table 2.3: Review of existing datasets' usage in machine learning based IDS between 2014 and 2018 (cnt'd)

Approaches	Dataset	Year
Random Forest+RandomTree+J48 [89]	AWID-CLS-R	2017
Bagging+MLP+Naïve Bayes [90]	AWID-ATK-R	2018
AdaBoost+logitBoost+ZeroR [91]	AWID-ATK-R	2018
Variational AE+Staged Classifiers+Majority voting [92, 93]	CICDS-001	2018
Deep learning [94]	AWID-CLS-R	2016
Random Tree +RandomForest [95]	AWID-ATK-R	2017
PSO+Association rules [96]	AWID-CLS-F	2017
Fully Unsupervised Deep Learning [50]	AWID-CLS-R	2017
Semi-supervised Deep Learning [97]	AWID-CLS-R	2016
Features selection+SVM+CFS+Corr+ANN [5, 98]	AWID-CLS-R	2017
PSO+SVM [99]	AWID-CLS-R	2017
Multi agent ANN [70]	AWID-CLS-R	2017
LibSVM+J48+Random Forest+Logistic [100]	GPRS	2016
Gradient boosted machine [77]	GPRS+UNSW-NB15	2017
Multi Level Classifier [101]	GPRS	2017
KNN+RF+ID3+Adaboost+ MLP+Naïve-Bayes+QDA [102]	GPRS	2018
KNN [103]	CIDDS-001	2018
PCA [104]	UGR'16	2017

Table 2.4: AWID-ATK-R dataset

AWID-ATK-R					
AWID-ATK-R-Trn			AWID-ATK-R-Tst		
Count	Type	Percentage	Count	Type	Percentage
1633190	normal	92%	477	amok	$< 1\%$
31180	amok	2%	13644	arp	3%
64609	arp	4%	599	beacon	$< 1\%$
3500	authentication_request	$< 1\%$	379	cafe_latte	$< 1\%$
1799	beacon	$< 1\%$	2871	chop_chop	1%
45889	cafe_latte	2%	1759	cts	$< 1\%$
10447	deauthentication	$< 1\%$	4445	deauthentication	1%
2633	evil_twin	$< 1\%$	84	disassociation	$< 1\%$
770	fragmentation	$< 1\%$	611	evil_twin	$< 1\%$
1558	probe_response	$< 1\%$	167	fragmentation	$< 1\%$
			19089	hirte	3%
			530785	normal	92%
			165	power_saving	$< 1\%$
			369	probe_request	$< 1\%$
			199	rts	$< 1\%$

environment. Wi-Fi is expected to increase sharply in the coming years as it is being used for not only wireless local area connectivity, but also for Internet of Thing (IoT) connectivity. Thus, continued efforts are required to make Wi-Fi environments more

secure. This section lists all the attack types against 802.11 protocols [90] that are incorporated in AWID, and provides a brief synopsis of the relevant attacks that our IDS can detect.

According to Kolias et al. [2], attacks against the 802.11 protocols can be divided into three main categories depending on the purpose, target, and methodology.

On the basis of attacker purpose: Key Cracking, Keystream, Denial of Service (DoS) and M-i-M (Man in the Middle) can be listed.

According to attacker target: client and network are the categories.

In terms of methodology, there are passive, injection, flooding, and impersonation attacks. In key retrieving sub-categories, the attacker tries to reveal the secret key through a monitoring process of certain packets in the flow of the network traffic. Then, the attacker proceeds in an offline manner using a key cracking process. Examples of these types of attacks are FMS, KeroK family, PTW, ARP injection, and Dictionary attacks. Moreover, examples of Keystream retrieving attacks are Chop-chop, Fragmentation, Caffe-Latte and Hirte attacks. These attacks are one of the many classes labeled in the AWID-ATK-R that we utilized in our research. Chop-chop is an attack that was originally implemented against the WEP key. In this attack, the attacker captures an ARP packet in an attempt to predict the key stream. Then, it uses the predicted key to forge a new packet which can later be injected into the network traffic to generate new Initialization Vectors (IVs). The evil twin is an active attack that works against the wireless access point. In this attack, the attacker sets up a rogue access point to be a legitimate wireless access point. Here, the rogue access point gathers privileged and corporate information without the knowledge of the end user. The cafe latte attack is an active attack. In this attack, the attacker that is in the radio frequency range of the legitimate network retrieves a WEP key from a client user. Here, the attacker captures an Address Resolution Protocol (ARP) packet from

Table 2.5: AWID data attacks description

Attack	Attacker Action
ARP -Injection	The attacker cracks the WEP key in an attempt to inject false Address Resolution Protocol (ARP) messages into the network
Chop-Chop	The attacker retrieves the last m bytes of the plaintext packet and key stream without having knowledge of the WEP key
Fragmentation	The attacker reveals a significant portion of the key stream and inject packets using the key stream
Caffe Latte	The attacker obtains the WEP key from a remote client using the ARP response
Hirte	The attacker retries the WEP key using an ARP message and without an AP of the network at all
Disassociation	The attacker transmits forged disassociation frames leading to connectivity loss
Deauthentication Broadcast	The attacker keeps transmitting forged Deauthentication broadcast frames in its vicinity that results in loss of the connectivity
Disassociation Broadcast	The attacker transmits forged disassociation frames to cause loss of the connectivity
Authentication Request	The attacker attempts to exhaust the AP's resources in an attempt to cause an overflow in the association table of the client
CTS	The attacker transmits the Clear To Send frames constantly and as a result the node will postpone the routing of messages
RTS	The attacker transmits the Ready To Send frames constantly forcing the node to defer the routing messages
Beacon	The attacker utilizing a fake Access Point emits many fake beacon signals that lead to Denial Of Service because the user cannot connect to the legitimate Access Point
Probe Response	The attacker floods the probe packets to exhaust the resources of the Access Point and the client
Evil Twin	The attacker brings up a fake AP that advertises the existing Extended Service Set Identification (ESSID) to a neighbor in the same frequency range

the isolated client. It then manipulates the forged packet and sends it back to the client. The fragmentation attack exploits the fragmentation process in the network to perform the attack. Fragmentation attack is similar to chop-chop attack. In fragmentation attack, the attacker tries to recover the WEP key by capturing a packet from the access point. Next, it will use this packet to perform the attack. Table 2.5 provides a brief description of each attack encompassed in the AWID-ATK-R dataset.

2.5.2 Relevant ML-Based IDS Work for AWID Dataset

To start, Kaleem et al. [105] proposed an agent-based malicious detection model that uses Artificial Neural Network (ANN) for the detection process. The authors applied their model on the AWID-CLS-R subset to classify each instant either as normal or a threat. It has been shown that the model provides highly accurate results (99.3%).

Thing [106] also used the AWID-CLS-R dataset, which has 4 classes. The

author considered a multi-class classification utilizing a deep learning approach that achieved an overall accuracy of 98.67% using 154 Features.

Kolias et al. [2] applied 8 conventional supervised machine learning classifiers to perform the attack classification on the AWID-CLS-R subset. They used the AdaBoost, J48, Naïve Bayes, OneR, Random Forest, Random Tree and ZeroR algorithms. The authors carried out manual feature selections and incorporated 20 features to train the classifiers. The overall accuracy of their classifiers ranged from 89.43% to 96.2%.

Aminanto et al. [97] proposed a framework for detecting active attacks using a Stacked Auto Encoder (SAE), which is an unsupervised learning method for feature selection. The framework used regression layer, a supervised learning technique, with SoftMax activation function for the classification process. The highest accuracy was 97.7% with 4 classes and 154 features. Moreover, Aminanto et al. [5, 98] employed three machine learning methods on AWID-CLS-R to select the best features to improve the detection process of impersonation attacks. The authors have eliminated 2 classes out of 4 classes of the AWID-CLS-R and kept two classes which are impersonation attacks and normal traffic classes. They employed Support Vector Machine (SVM), Artificial Neural Network (ANN), and Decision Tree. ANN was used for the attack classification. The results show that their approach was 99.86% accurate in detecting impersonation attacks.

Thanthrige et al. [89, 95] applied 5 supervised machine learning classifier algorithms to perform the attack classification on the AWID-CLS-R and AWID-ATK-R subsets. The algorithms they used included AdaBoost, J48, OneR, Random Forest and Random Tree. The selected features were evaluated and ranked by using Information Gain and Chi-Square measures. The classifiers were applied on AWID-ATK-R [2] and AWID-CLS-R [2] with different settings of features based on the feature evalu-

ation results. The highest achieved accuracy was 95.12% using Random Tree applied on AWID-CLS-R with a total number of 41 features. In addition, the highest achieved accuracy was 94.97% using the Random Forest classifier applied on AWID-ATK-R using the same number of features. The results show that reduced number of features (41 features) enhances the accuracy. However, the accuracy decreased when the authors further reduced the features to 10.

Kolias et al. proposed TermID [96], which is a distributed network intrusion detection system. TermID uses Classification Rule Induction (CRI) and Swarm Intelligence Optimization (SIP) in an attempt to achieve efficient model training, without exchanging sensitive data. This system includes: two operational units, the monitor nodes and the central node. The monitor nodes transform the input examples from their local dataset to intermediate summaries, while the central node performs reduce operations on the global dataset and runs the main body of the rule construction process. TermID[96] used the AWID-ATK-F dataset that was manually broken down and distributed to each node. The authors did not report accuracy. Table 2.6 provides a summary of the related work.

Table 2.6: Summary of work related to AWID

Study	Dataset	Approach	Attributes (Features)	Classes	Accuracy (%)
Kolias [2]	AWID-CLS-R	J48	20	4	96.2
Aminanto [98]	AWID-CLS-R	ANN	154	2	99.86
Aminanto [97]	AWID-CLS-R	SAE	154	4	97.7
Kaleem [105]	AWID-CLS-R	ANN	7	2	99.3
Thanthrige [89]	AWID-CLS-R	Random Forest	111	4	94.83
Thanthrige [89]	AWID-CLS-R	Random Tree	41	4	95.12
Thanthrige [89]	AWID-CLS-R	J48	10	4	92.44
Thing [106]	AWID-CLS-R	Deep Learning	154	4	98.67
Thanthrige [95]	AWID-ATK-R	Random Tree	111	4	94.58
Thanthrige [95]	AWID-ATK-R	Random Forest	41	4	94.97
Thanthrige [95]	AWID-ATK-R	Random Forest	10	4	92.29

2.6 CICIDS2017 Dataset Structure

The CICIDS2017 dataset [8] consists of realistic background traffic that represents the network events produced by the abstract behavior of a total of 25 users. The users' profiles were determined to include specific protocols such as HTTP, HTTPS, FTP, SSH and email protocols. The developers used statistical metrics such as minimum, maximum, mean and standard deviation to encapsulate the network events into a set of certain features which include:

(1) The distribution of the packet size

(2) The number of packets per flow

(3) The size of the payload

(4) The request time distribution of the protocols

(5) Certain patterns in the payload

Moreover, CICIDS2017 covers various attack scenarios that represent common attack families. The attacks include Brute Force Attack, HeartBleed Attack, Botnet, DoS Attack, Distributed DoS (DDoS) Attack, Web Attack, and Infiltration Attack.

The dataset is publicly available by the authors in two formats:

(1) The full packet payloads in Packet CAPture (PCAP) format

(2) The corresponding profiles and labeled flows as CSV files for machine and deep learning purposes

2.6.1 Attacks in CICIDS2017

CICIDS2017 was collected based on real traces of benign and malicious activities of the network traffic. The total number of records in the dataset is 2,830,108.

The benign traffic encompasses 2,358,036 records (83.3% of the data), while the malicious records are 471,454 (16.7% of the data). CICIDS2017 is one of the unique datasets that includes up-to-date attacks. Furthermore, the features are exclusive and matchless in comparison with other datasets.

Table 2.7 highlights the characteristics and distribution of the attacks in the CICIDS2017 dataset and provides a brief description of each type of attack. CICIDS2017 is a labeled dataset with a total number of 84 features including the last column corresponding to the traffic status (class label). The features were extracted by CICFlowMeter-V3 [107]. The output of CICFlowMeter-V3 is a CSV file that includes: Flow ID (1), Source IP (2) and Destination IP (4), Time stamp (7) and Label (84). The Flow ID (1) includes the four tuples: Source IP, Source Port, Destination IP, and Destination Port. Time stamp represents the timing. To the best of our knowledge, all previous studies that used CICIDS2017 neglect Flow ID (1), Source IP (2), Destination IP (4), and Time stamp (7). In this work, we used CICIDS2017 with respect to the listed features except the Flow ID (1) and Time Stamp (7). Thus, in our study, the total number of used features encompasses 82 features including the Label (84). These features are listed in Table 2.8. The extracted traffic features are explained in [108].

2.6.2 Relevant ML-Based IDS Work for CICIDS2017 Dataset

This section provides a brief overview of the prior related work that are directly relevant to CICIDS2017, with a special emphasis on machine learning approaches that utilize this dataset. It also highlights a sample of previous work that utilize feature reduction methods in machine learning based IDS.

Sharafaldin et al. [8] used a Random Forest Regressor to determine the best set of features to detect each attack family. Then, the authors examined the performance

Table 2.7: CICIDS2017 characteristics, distribution and brief description of each type of attack

Traffic Type	Size	Description
Benign	2,358,036	Normal traffic behavior
DoS Hulk	231,073	The attacker employs the HULK tool to carry out a denial of service attack on a web server through generating volumes of unique and obfuscated traffic. Moreover, the generated traffic can bypass caching engines and strike a server's direct resource pool
Port Scan	158,930	The attacker tries to gather information related to the victim machine such as type of operating system and running service by sending packets with varying destination ports
DDoS	41,835	The attacker uses multiple machines that operate together to attack one victim machine
DoS GoldenEye	10,293	The attacker utilizes the GoldenEye tool to perform a denial of service attack
FTP Patator	7,938	The attacker uses FTP Patator in an attempt to perform a brute force attack to guess the FTP login password
SSH Patator	5,897	The attacker uses SSH Patator in an attempt to perform a brute force attack to guess the SSH login Password
DoS Slow Loris	5,796	The attacker uses the Slow Loris tool to execute a denial of service attack
DoS Slow HTTP Test	5,499	The attacker exploits the HTTP Get request to exceed the number of HTTP connections allowed on a server, preventing other clients from accessing and giving the attacker the opportunity to open multiple HTTP connections to the same server
Botnet	1,966	The attacker utilizes trojans to breach the security of several victim machines, taking control of these machines and organizes all machines in the network of Bot that can be exploited and managed remotely by the attacker
Web Attack: Brute Force	1,507	The attacker tries to obtain privilege information such as password and Personal Identification Number (PIN) using trial-and-error
Web Attack: XSS	625	The attacker injects into otherwise benign and trusted websites using a web application that sends malicious scripts
Infiltration	36	The attacker utilizes infiltration methods and tools to infiltrate and gain full unauthorized access to the networked system data
Web Attack: SQL Injection	21	SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution
HeartBleed	11	The attacker exploits the OpenSSL protocol to insert malicious information into OpenSSL memory, enabling the attacker with unauthorized access to valuable information

Table 2.8: Listed features of network traffic in CICIDS2017.

No.	**Feature**	**No.**	**Feature**	**No.**	**Feature**
1	Flow ID	29	Fwd IAT Std	57	ECE Flag Count
2	Source IP	30	Fwd IAT Max	58	Down/Up Ratio
3	Source Port	31	Fwd IAT Min	59	Average Packet Size
4	Destination IP	32	Bwd IAT Total	60	Avg Fwd Segment Size
5	Destination Port	33	Bwd IAT Mean	61	Avg Bwd Segment Size
6	Protocol	34	Bwd IAT Std	62	Fwd Avg Bytes/Bulk
7	Time stamp	35	Bwd IAT Max	63	Fwd Avg Packets/Bulk
8	Flow Duration	36	Bwd IAT Min	64	Fwd Avg Bulk Rate
9	Total Fwd Packets	37	Fwd PSH Flags	65	Bwd Avg Bytes/Bulk
10	Total Backward Packets	38	Bwd PSH Flags	66	Bwd Avg Packets/Bulk
11	Total Length of Fwd Pck	39	Fwd URG Flags	67	Bwd Avg Bulk Rate
12	Total Length of Bwd Pck	40	Bwd URG Flags	68	Subflow Fwd Packets
13	Fwd Packet Length Max	41	Fwd Header Length	69	Subflow Fwd Bytes
14	Fwd Packet Length Min	42	Bwd Header Length	70	Subflow Bwd Packets
15	Fwd Pck Length Mean	43	Fwd Packets/s	71	Subflow Bwd Bytes
16	Fwd Packet Length Std	44	Bwd Packets/s	72	Init_Win_bytes_fwd
17	Bwd Packet Length Max	45	Min Packet Length	73	Act_data_pkt_fwd
18	Bwd Packet Length Min	46	Max Packet Length	74	Min_seg_size_fwd
19	Bwd Packet Length Mean	47	Packet Length Mean	75	Active Mean
20	Bwd Packet Length Std	48	Packet Length Std	76	Active Std
21	Flow Bytes/s	49	Packet Len. Variance	77	Active Max
22	Flow Packets/s	50	FIN Flag Count	78	Active Min
23	Flow IAT Mean	51	SYN Flag Count	79	Idle Mean
24	Flow IAT Std	52	RST Flag Count	80	Idle Packet
25	Flow IAT Max	53	PSH Flag Count	81	Idle Std
26	Flow IAT Min	54	ACK Flag Count	82	Idle Max
27	Fwd IAT Total	55	URG Flag Count	83	Idle Min
28	Fwd IAT Mean	56	CWE Flag Count	84	Label

of these features with different algorithms that included K-Nearest Neighbor (KNN), Adaboost, Multi-Layer Perceptron (MLP), Naïve Bayes, Random Forest (RF), Iterative Dichotomiser 3 (ID3) and Quadratic Discriminant Analysis (QDA). The highest

precision value was 0.98 with RF and ID3 [8]. The execution time (time to build the model) was 74.39 seconds. This is while the execution time for our proposed system using Random Forest is 21.52 seconds with a comparable processor. Furthermore, our proposed intrusion detection system targets a combined detection process of all the attack families.

In wireless mesh environments, Vijayan et al. [109] proposed an intrusion detection system that utilized the genetic algorithm (GA) as a feature selection method and multiple Support Vector Machines (SVM) for classification. Their system was based on a linear combination of multiple SVM classifiers, which were ordered based on the severity of the attacks. Each classifier was trained to detect a certain attack category using selected features by the GA. A small portion of the CICIDS2017 dataset instances were used to evaluate their system. Conversely, we use all the instances of the CICIDS2017 dataset.

Moreover, authors in [110] compared and contrasted a frequency-based model from five sequence of aggregation rules with sequence-based modeling of the Long Short-Term Memory (LSTM) recurrent neural network. The investigation concluded that the frequency-based model tends to perform similar or better than the LSTM models in detecting the attacks.

Additionally, the researchers in [111] analyzed the CICIDS2017 dataset using digital wavelets. Their method efficiently detected service denial attacks of both Slow Loris and HTTP Denial of Service (DoS).

Furthermore, the authors of [112] applied the Multi-Layer Perceptron (MLP) classifier algorithm and a Convolutional Neural Network (CNN) classifier that utilized the Packet CAPture (PCAP) file of CICIDS2017. The authors selected specified network packet header features for the purpose of their study. Conversely, we utilized the corresponding profiles and the labeled flows for machine and deep learning purposes.

According to [112], the results demonstrated that the payload classification algorithm was judged to be inferior to MLP. However, it showed significant ability to distinguish network intrusion from benign traffic with an average true positive rate of 94.5% and an average false positive rate of 4.68%.

The authors in [113] proposed a denial of service intrusion detection system that utilized the Fisher Score algorithm for features selection and Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Decision Tree (DT) as the classification algorithm. Their IDS achieved 99.7%, 57.76% and 99% success rates using SVM, KNN and DT, respectively. In contrast, this research proposed an IDS to detect all types of attacks embedded in CICIDS2017 and as shown in Figure 5.10, achieves 100% accuracy for DDoS attacks using $(PCA - RF)_{Mc-10}$ with UDBB. The authors in [114] utilized a distributed Deep Belief Network (DBN) as the the dimensionality reduction approach. Then, the obtained features were fed to a multi-layer ensemble SVM. The ensemble SVM is accomplished in an iterative reduce paradigm based on Spark (which is a general distributed in-memory computing framework developed at AMP Lab, UC Berkeley), to serve as a Real Time Cluster Computing Framework that can be used in big data analysis [115]. Their proposed approach achieved a value of F-measure equal to 0.921.

The authors in [116] proposed a Data Dimensionality Reduction (DDR) method. Their proposed scheme was evaluated by XGBoost (Extreme Gradient Boosting) [117], SVM (Support Vector Machine), CTree (Conditional inference Trees) [118] and Neural network (Nnet) classifiers. The number of selected features was 36 and the highest achieved accuracy was 98.93% with XGboost. Furthermore, the authors excluded Monday network traffic, which is only benign traffic in their system. This is while our work was able to achieve accuracy of 99.% with 10 features. In addition, we kept all the files of the dataset that represent different classes of the network traffic.

Table 2.9: Summary of previous work related to CICIDS2017

Reference	Classifier name	F-measure	Feature selection/extraction (Features Count)
[8]	KNN RF ID3 Adaboost MLP Naïve Bayes QDA	0.96 0.97 0.98 0.77 0.76 0.04 0.92	Random Forest Regressor (54)
[112]	MLP	0.948	Payload related features
[114]	SVM	0.921	DBN
[109]	GA+SVM	0.971	GA (40)
[113]	KNN	0.997	Fisher Scoring (30)
[119]	XGBoost for DoS Attacks	0.995	(80)
[120]	Deep Learning for Port Scan Attacks	Accuracy 97.80	(80)
[120]	SVM for Port Scan Attacks	Accuracy 69.79	(80)
[116]	XGBoost	Accuracy 98.93	DDR Features Selections (36)
[121]	Deep Multi Layer Perceptron (DMLP) for DDoS Attacks	Accuracy 91.00	Recursive feature elimination with Random Forest

2.7 CIDDS-001 Dataset Structure

The CIDDS-001 (Coburg Intrusion Detection Dataset) [7] is a labeled flow based dataset that was released in 2017 in a cloud environment based on the OpenStack platform. CIDDS-001 contains unidirectional NetFlow and was created using an emulated small business environment which included web-server, E-mail server, and groups of variant clients. CIDDS-001 encompasses benign and malicious traffic. The benign traffic simulated a realistic user behavior environment that considers individual working schedule with various working tasks [122]. The malicious traffic includes Denial of Service (DOS), Brute Force, and port scan attacks. The first 10 attributes (features), as tabulated in Table 2.10, are the default NetFlow attributes, and the last four attributes are additional attributes (description attributes).

Table 2.10: CIDDS-001 features set

Number	Name	Description
1	Src IP	Source IP Address
2	Src Port	Source Port
3	Dest IP	Destination IP Address
4	Dest Port	Destination Port
5	Proto	Transport Protocol (e.g. ICMP, TCP, or UDP)
6	Date first seen	Start time flow first seen
7	Duration	Duration of the flow
8	Bytes	Number of transmitted bytes
9	Packets	Number of transmitted packets
10	Flags	OR concatenation of all TCP Flags

2.7.1 Attacks in CIDDS-001

The CIDDS-001 dataset includes 43 attacks. 20 attacks were executed in week 1 and 23 attacks were executed in week 2. Week one and week two contain traffic with benign behavior as well as attacks. Table 2.11 provides more information about the executed attacks within the CIDDS-001 dataset [122].

Table 2.11: CIDDS-001 attack types

Traffic Status
Normal
SYN Scan
ACK Scan
UDP Scan
FIN Scan
Ping Scan
Port Scan
Brute Force
Dos

2.7.2 Relevant ML-Based IDS Work for CIDDS-001 Dataset

This section provides a brief overview of prior related IDS work with a special emphasis on deep and machine learning approaches that utilized CIDDS-001 as the benchmark dataset.

In the Internet of Things (IoT) environments, Tama et al. [123] used deep neural networks which combined the grid search strategy to classify attacks. Tama and Rhee [123] utilized 10 fold cross-validation (10-FCV); repeated cross-validation of 5 repetitions each with 2-FCV approaches; in the classification process. The authors split the data into 5 sub-samples and validated these based on 2-FCV. The sub-samples were randomly chosen. Thus, the approach may use the same samples that belong to the majority class of the dataset repeatedly. We consider the class distribution of instances when applying sampling techniques in our work. This is while Tama and Rhee's validation methods [123] did not significantly affect the performance of their proposed approach.

For Traditional Information Communication System (TICS) environments, He et al. [124] proposed a Denial of Service (DOS) attack detection system that utilized nine classification techniques in supervised and unsupervised scenarios. For the supervised cases, Decision Tree (DT), Naïve Bayes, Random Forest (RF), and Support Vector Machine (SVM) with variant kernel types such as, linear, poly and Radial Basis Function (RBF) were used. For the unsupervised scenarios, the authors used K-means classification. The authors extracted the statistical features of four Distributed Denial of Service (DDoS) attacks that included SSH Brute-force, Dynamic Domain Server (DNS) Reflection, Internet Control Message Protocol (ICMP) Flood, and Transport Control Protocol (TCP) SYNchronize SYN. They further evaluated the performance of their proposed system through launching real attacks in lab settings. The experimental results showed that their system was able to detect attacks

with an accuracy of 99.7% and low false positives of 0.07.

In cloud environments, Idhammad et al. [125] developed a HyperText Transfer Protocol (HTTP) DDoS attacks detection system based on Information Theoretic Entropy and Random Forest. The authors estimated the entropy of the network header features using a time-based sliding window algorithm. In the same manner, Idhammad et al. [126] proposed a distributed intrusion detection system (IDS) based on machine learning approaches that utilized the Random Forest classifier to detect attacks in cloud environments. Their IDS incorporated 5 modules that included a network traffic module to capture the incoming network traffic on the edge network routers on a 5-minute time window basis. The suspected traffic at each edge network router was synchronized to the central server. Then, an ensemble learning classifier based on Random Forest was used to classify the network traffic on the central storage server and detect the type of each attack. The authors implemented the system in Google cloud platform and tested it using the CIDDS-001 dataset.

Nicholas et al. [127] evaluated the performance of Long Short-Term Memory (LSTM) utilizing CIDDS-001. They compared the LSTM performance with various machine learning classifiers.

Verma and Ranga [128] utilized the k-nearest neighbor (KNN) classifier on CIDDS-001 to build an IDS. Their system achieved an overall accuracy of 99.6% with 2NN and a minimum accuracy of 99.3% with 5NN.

Researchers have not dealt with or treated the imbalanced class distributions of CIDDS-001 in much detail. The key contributions of this work include highlighting the importance of imbalanced class problems in network traffic for devising an effective IDS, and studying the effect of various sampling techniques on the CIDDS-001 dataset using different classification approaches.

Imbalanced classes; where a particular class is over-represented; can occur in

Table 2.12: Summary of work related to CIDDS-001

Study	Approach	Performance
Tama et al. [123]	Deep NN	92.5%
He et al. [124]	RF, SVM, DT	99.7%
Idhammad et al. [125]	RF+Entropy	98.68%
Nicholas et al. [127]	LSTM	97.8%
Verma and Ranga [128]	K+2NN	99.6%

many data fusion problems that target data classification [92]. This dissertation provides novel insights on the imbalanced class distribution concerns in data mining of big data and suggests approaches on how to tackle this important issue.

2.8 Relevant Imbalanced Class Work in IDS

There is a relatively small body of literature that is concerned with imbalanced class problems in intrusion detection systems.

Rodda and Erothi [129] analyzed the effect of imbalanced class problems on the NSLKDD benchmark dataset [130] utilizing Random Forest, J48, Naïve Bayes and BayesNet classification techniques. The authors did not adopt any of the distribution balancing methods in their work. This is while in our work, 4 methods are adopted.

In the same manner, Balasubramanian and Joseph [131] studied the affect of the highly imbalanced NSLKDD dataset. The authors applied two data managing/balancing methods; namely re-sample with replacements (RWR) and re-sample without replacements (RWoR). On the other hand, this work adopts the latter and further applied SMOTE and Class Balancing methods.

Cieslak et al. [132] used the RIPPER classifier and dataset to generate two imbalanced datasets with SNORT. The authors implemented a combination of oversampling and under-sampling techniques to combat the imbalanced distribution problem in the collected data. Furthermore, Cieslak et al. [132] applied k-means clustering to

the minority class followed by SMOTE to each cluster. In contrast, this work used a benchmark dataset (AWID) and applied SMOTE at once to each of the minority classes. Furthermore, this work applied the class balancing method as well.

Teshome and Rao [133] examined cost-sensitive methods on the imbalanced KDDCUP99 and NSLKDD datasets utilizing CS-MC4 and CS-CRT classifiers. This is while this work examined sampling, synthetic sampling and balancing methods on the up-to-date Wi-Fi benchmark dataset in addition to Boosting, multi-class classification (one against all) using Naïve Bayes, and Random Forest classification methods.

Engen [134] evolved the weights of an Evolutionary Neural Network (ENN) using Genetic Algorithms (GA). Moreover, to accommodate the classification trade-off of the Evolutionary Neural Network, the authors applied Multi-Objective GA (MOGA), which treats the classification rate on each class as a separate objective.

To deal with the imbalanced classes in the NSLKDD, Parsaei et al. [135] developed a hybrid approach that incorporated a combination of SMOTE, Cluster Center and Nearest Neighbor (CANN) to select the effective features along with a leave one out method (LOO).

Khor et al. [136] utilized under-sampling and oversampling methods to mitigate the skewed class problems in NSLKDD. Khor [136] under-sampled the majority classes and used SMOTE for oversampling the minority classes. Then, the authors evaluated the resulted data using Naïve Bayes, Bayesian Networks, ID3, C4.5, and CART classifiers.

Kotsiantis et al. [137] reviewed various techniques for handling imbalanced datasets and the relationship between the training set sizes. They concluded that the minority class is poorly represented for small imbalanced datasets and is not sufficient for learning, especially when a large degree of class overlapping exists.

Yan et al. [138] proposed a Region Adaptive Synthetic Minority Oversampling

Technique (RA-SMOTE) to solve the imbalanced problems in NSLKDD. The authors tested RA-SMOTE using Support Vector Machines (SVM), Back-propagation Neural Network (BPNN), and Random Forests (RF). A summary of previous work related to combating imbalanced class distributions is presented in Table 2.13.

Table 2.13: Summary of work related to imbalanced class problems in IDS

Study	Dataset	Approaches
Rodda and Erothi [129]	NSL-KDD	Classification
Balasubramanian and Joseph [131]	NSL-KDD	RWR, RWoR
Cieslak et al.[132]	Synthetic dataset of Snort	RWR, RWoR ,K-mean+SMOTE
Teshome and Rao [133]	KDDCUP99, NSLKDD	Classification (CS-MC4,CS-CRT)
Engen [134]	KDDCUP99	Classification (MOGA)
Parsaei et al.[135]	NSL-KDD	SMOTE+CANN
Khor et al. [136]	NSL-KDD	RWR, RWoR, SMOTE
Kotsiantis et al. [137]	NSL-KDD	RWR, RWoR
Yan et al. [138]	NSL-KDD	RA-SMOTE
Tama et al. [123]	CIDDS-001	10 and 2 fold cross validation

2.9 Cyber Physical Systems (CPS)

Cyber Physical Systems (CPS) [139] are configured from the integration of computational algorithms and physical components. Such CPS systems include large public infrastructures such as water treatment plants, smart grids, and transportation system, as well as physically smaller systems or devices such as pacemakers or insulin pumps [140]. Given the potential for cyber-attacks, it is prudent to design mechanisms for detecting and defending a CPS against either cyber-attack or physical attacks. A cyber-attack attempts to disturb the communication network of a CPS, whereas a physical attack is an attempt to tamper with the physical elements of a CPS such as insulin sensor and pumps, etc. This section of the dissertation is focused on Medical Cyber Physical Systems (MCPS) that monitor the patient's health through various medical devices.

2.10 Relevant CPS Work

Behavior-specificationâbased intrusion detection is a form of behavior-based intrusion detection that does not leverage user, group, or data profiling within a system. Instead, security system experts create rules that reflect legitimate behaviors of a system. Here, an IDS will measure a node's misbehavior according to the node's deviation from the specified rules of a system. This process allows for lightweight intrusion detection to be deployed into systems with severe resource constraints where data profiling is difficult. Specification is a set of rules and thresholds that define the expected behavior for network components such as nodes, protocols, and routing tables. Specification-based approaches detect intrusions when network behavior deviates from specification definitions. Several attempts have been made to build an intrusion detection system that applies specification-based rules to detect intruders.

In 2005, DaSilva et al. [47] proposed an IDS that used seven types of traffic-based rules to incorporate interval, integrity, and delay parameters to detect intruders.

Cheung et al. [141] studied a behavior specificationâbased IDS that used the Prototype Verification System (PVS) to transform protocol, communication pattern, and service availability specifications into a format compatible with EMERALD [142] and Snort [143]. The authors audited the fields of Modbus packets. In particular, Cheung et al. ensured individual fields fall within the valid ranges and relationships between fields are preserved. This study lacks any numerical results.

By 2011, Carcano et al. [144] introduced an IDS that applied critical state analysis and state proximity to monitor the evolution of the targeted system's states to detect intruders. Furthermore, Mitchell and Chen [145] proposed IDS's for aerospace and smart utility (power) applications, respectively. These are all behavior-specification-based approaches driven by a state machine based upon behavior rules created by

security experts.

Additionally, Li et al. [146] built an intrusion detection system for a smart grid CPS that factored behavior rules of line meter devices. They monitored the device's behaviors to determine whether the smart grid CPS violated the normal behavior.

In works related to MCPS, Asfaw et al. [147] proposed an anomaly-based IDS for these systems. The authors focused on attacks that violated privacy of the targeted systems; whereas, in our case study experiment, we focused on attacks that violated the integrity of MCPS. Asfaw et al. [147] used an anomaly-based approach while we applied a specification-based rules approach. Also, Asfaw et al. [147] did not provide numerical results in the form of false negatives or positives.

Yang and Hwang [148] introduced a fraud and abuse detection approach for healthcare applications. Our research, on the other hand, focuses on the treatment, rather than the administrative domain of healthcare. Furthermore, Yang and Hwang used an anomaly-based approach, while we used a specification-based rules approach. They provided numerical results that measured the internal validity. However, they did not provide externally valid metrics such as the Receiver Operating Characteristic (ROC).

Furthermore, Mitchell and Chen [149] proposed an IDS for medical applications. The authors used behavior-specification-based approaches driven by a state machine derived from behavior rules constructed by security experts. Moreover, they considered seven threshold monitoring approaches: Binary, Hamming, Manhattan, Euclidean, LCS, Levenshtein, and Damerau-Levenshtein. The authors created a dataset for good nodes and bad nodes using Monte Carlo simulation. Mitchell and Chen assumed the nodes were corrupted from several types of attackers such as reckless and random attackers. Mitchell and Chen considered command injection, greyhole and exfiltration attacks.

CHAPTER 3: METHODOLOGY AND MILESTONES

3.1 Machine Learning-Based IDS

In the literature, there are a number of intrusion detection systems that are developed based on machine learning techniques. Machine learning techniques can predict and detect threats before they result in major security incidents. In general, the structure of a machine learning IDS, as highlighted in Figure 3.1 is composed of Pre-processing, Features reduction/selection, Classification and Evaluation. One of

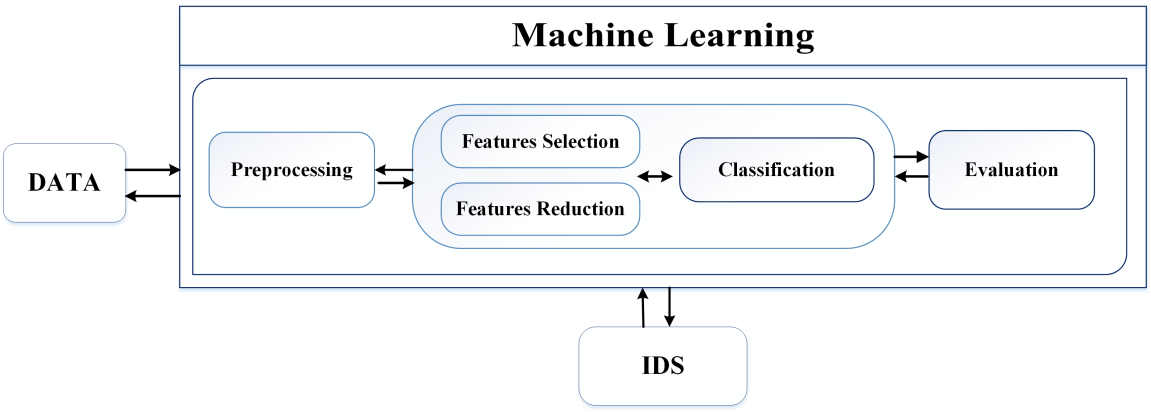

Figure 3.1: Generic structure of Machine Learning-Based IDS

the most well-known methods in machine learning for assessing the detection process in various domains is classification. Classification can be defined as the process of grouping or categorizing an object according to shared characteristics and features.

Classification is either supervised or unsupervised. When the dataset is labeled, then the preferred classification techniques are the supervised ones. To be more specific, some studies apply single learning techniques, such as random forest, multilayer perceptron, support vector machines and genetic algorithms. On the other hand, some systems are based on combining different learning techniques, such as voting, stacking and ensemble techniques [150]. In particular, these techniques are developed as classifiers, which are used to classify or recognize whether the incoming traffic is the normal traffic or malicious. Some systems employ feature reduction and/or feature selection prior to classification. As a result, the performance of an IDS is significantly improved when the features are more discriminative and representative. Moreover, the huge network traffic and high-dimensional features lead to prolonged classification processes, whereas, low-dimensional features can shorten these processes.

3.2 Feature Selection

Feature selection is the most important procedure for any machine learning based IDS. Moreover, they are useful for reducing model complexity, which leads to faster learning and real-time processes.

From our literature review that is relevant to AWID Datatset [2], two of the reported intrusion detection systems applied manual features selection, whereas one study used Correlation based Feature Selection (CFS) [98]. Moreover, a study by Thanthrige used Chi-Square and Information Gain [89, 95]. Information Gain evaluates the worth of a feature by measuring the information gain with respect to the class, whereas Chi-Square evaluates the worth of a feature by computing the value of the chi-squared statistic with respect to the class.

Furthermore, Aminanto [98] used the C4.5 algorithm as a feature selection method. This method belongs to the family of decision-tree based algorithms and is

an extension to the well-known decision tree induction algorithm [151]. In addition, Aminanto [98] used the correlation attribute evaluator (Corr), which evaluates the worth of an attribute (feature) by measuring the its correlation with the class.

Figure 3.2 illustrates features selection methods, some, along with their classifiers, for existing IDS's that utilized the AWID dataset.

In our dissertation and for AWID [2], we use search algorithms such as Harmony search, Best First search, Bee search, and Ant Colony search in combination with feature set evaluators such as Cost sensitive and Correlation features subset evaluators. The following sections give a brief explanation of the feature selection techniques.

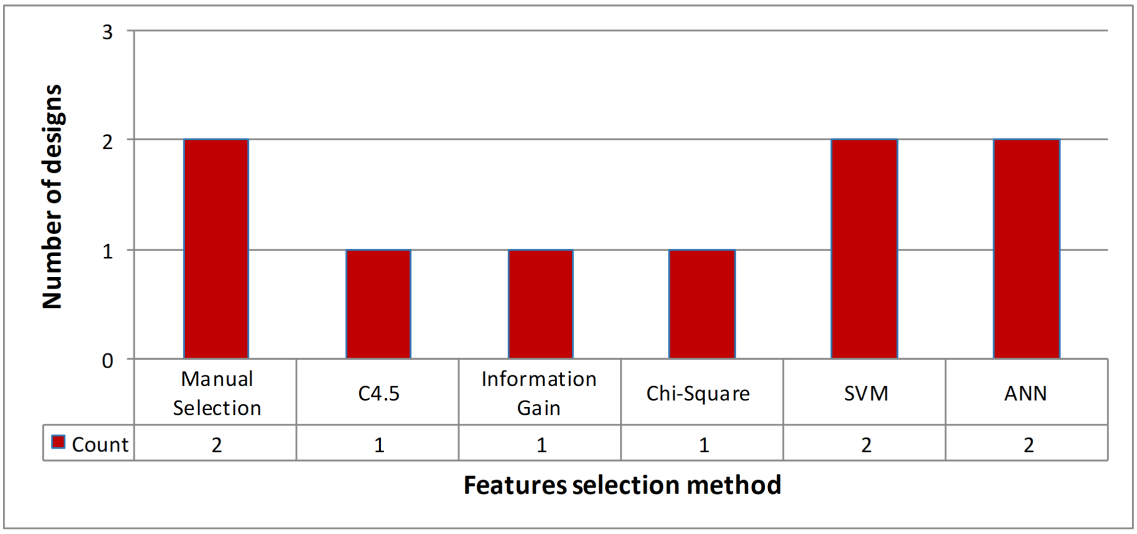

Figure 3.2: Statistics of relevant ML methods in AWID

3.2.1 Harmony Search (HS)

A well-known technique for effective features selection is the Harmony Search (HS) algorithm [152]. The Harmony Search (HS) algorithm is one of the meta-heuristic algorithms that mimics the improvisation process of music players [153]. One advantage of HS is that it avoids the problem of imposing complicated mathe-

matical requirements and it is not sensitive to the initial value settings [154] [155]. The HS method encompasses two phases: initialization and iteration. In the initialization phase, the HS parameters are defined, the memory is filled with random harmonics, and the harmonics are evaluated. In the iteration phase, a new harmony is improvised and evaluated. Based on the evaluation, either the new Harmony is discarded or it is replaced by the worst harmony in the harmony memory, and the harmony memory is therefore updated [152].

3.2.2 Ant-Colony Search

The Ant Colony Optimization algorithm [156] is a probabilistic technique that can be utilized as a feature selection technique which explores the features space [157, 158]. AntSerach uses two components to guide the search process for the pheromone value and the heuristic information.

3.2.3 Bee Search Algorithm

The Artificial Bee Colony, developed by D. Karaboga, is a Swarm Intelligence algorithm used to solve optimization problems in several research areas such as intrusion detection systems. Bee Search algorithm [159, 158] has the ability to explore the space of features and evaluate the subset of features.

3.2.4 Best First Search (BFS)

The Best First Search, searches the space of feature subsets by greedy hill climbing augmented with a backtracking facility [160]. Setting the number of consecutive non-improving nodes allowed, controls the level of backtracking done. The BFS may start with the empty set of features and search forward, or start with the full set of features and search backward, or start at any point and search in both directions

(through considering all possible single attribute/feature additions and deletions at a given point) [158].

3.2.5 Correlation Feature Subset Evaluate

The Correlation feature Subset Evaluate (CfsSubsetEval), evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low inter-correlation are preferred [25, 161].

3.3 Feature Reduction

For decades, researchers used dimensionality reduction approaches [162, 163, 164] for different reasons such as to reduce the computational processing overhead, reduce noise in the data, and for better data visualization and interpretation. One common dimensionality reduction approach is the Missing Value Ratio (MVR) approach [165]. The MVR approach is efficient when the number of missing values is high. For the CICIDS2017 dataset, the number of missing values is near zero. Therefore, we excluded the Missing Value Ratio approach. Other common approaches include the Forward Feature Construction (FFC) and Backward Feature Elimination (BFE) approaches [165]. Both FFC and BFE are prohibitively slow on high dimensional datasets, which is the case for CICIDS2017 (>2,500,500 instances). As a result, we did not discuss these approaches. The PCA technique, on the other hand, is relatively computationally cost efficient, can deal with large datasets, and is widely used as a linear dimensionality reduction approach [162, 166]. The auto-encoder dimensionality reduction approach is an instance of deep learning, which is also suitable for large datasets with high dimensional features and complex data representations [167].

This work adopts AE as well as PCA for features dimensionality reduction. One of the most fundamental differences between AE and PCA in terms of dimensionality reduction is that in the auto-encoder approach, there is no assumption of linearity in the data. The auto-encoder optimizer figures out the function through the weights that best encode the data under the specified reconstruction error metric. This is while the PCA assumes linearity in the set of reduced data [168].

3.3.1 Auto Encoder Based Feature Reduction

In this section, we present the sparse auto-encoder learning algorithm [169, 170], which is one approach to automatically learn feature reduction in unsupervised settings. Figure 3.3 shows the structure of the auto-encoder. The input vector $x = (x_1, x_2, ..., x_n)$ is first compressed to a lower dimensional hidden representation that consists of one or more hidden layers $a = (a_1, a_2, ..., a_m)$. The hidden representation a is then mapped to reproduce the output $\hat{x} = (\hat{x_1}, \hat{x_2}, ..., \hat{x_n})$. Let j be the counter parameter for the neurons in the current layer l, and i be the counter parameter for the neurons in the previous hidden layer $l-1$. The output of a neuron in the hidden layer can be represented by the following formula.

$$a_j^{(l)} = f(z_j^{(l)}) = f(\sum_{i=1}^{n} W_{ji}^{(l-1)}.a_i^{(l-1)} + b_j^{(l-1)}) \tag{3.1}$$

The size of the weight matrix of the hidden layer is represented by $W \in R^{m \times n}$ and the bias is $b \in R^m$. A sigmoid function is chosen as the activation function, such that $f(z) = \frac{1}{(1+exp^{-z})}$. Parameters W and b are optimized using back propagation, by minimizing the cost function J for all the training instances [171], as follows:

$$J(W, b; \hat{x}, x) = \frac{1}{k}\sum_{i=1}^{k}(\frac{1}{2}||\hat{x} - x||^2) + \frac{1}{\lambda}\sum_{l=1}^{ls-1}\sum_{j=1}^{m}\sum_{i=1}^{n}(W_{ji}^{(l)})^2 \tag{3.2}$$

Parameter λ is chosen to control the regularization term of all the weights in

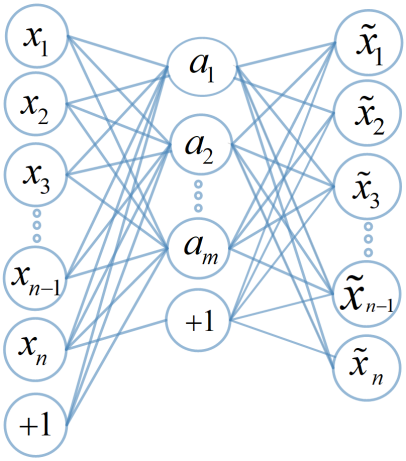

Figure 3.3: The structure of an AE.

a particular layer, and ls denotes the total number of layers. To impose a sparsity constraint on the hidden units, one strategy is to add an additional term in the loss function during training to penalize the Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ and a desired sparsity mean $\hat{\rho}_j$.

$$\hat{\rho}_j = \frac{1}{k} \sum_{i=1}^{k} \left[a_j^{(i)}(x^{(i)}) \right] \tag{3.3}$$

where $a_j^{(i)}$ denotes the activation of hidden unit j in the auto-encoder and k is the training sample [172].

$$J_{sparse}(W, b) = J(W, b; \hat{x}, x) + \beta \sum_{j=1}^{m} KL(\rho \,||\, \hat{\rho}_j) \tag{3.4}$$

This sparsity is guaranteed to have the effect of causing $\hat{\rho}_j$ to be close to ρ, because it ensures that the sparse activations are achieved on the training data for any given units in the hidden layer. The value of β is chosen to control the weight of the sparsity penalty term.

The computational complexity of executing the designed auto-encoder with a single hidden layer depends on the dimensionality of the input vector n, and the

Reduction ratio $R \in (0, 1)$ [173].

$$O(n.\ (R \times n) + (R \times n)\ .n) = O(Rn^2 + Rn^2) = O(n^2) \tag{3.5}$$

In this dissertation, a two hidden-layer sparse auto-encoder is used with sigmoid activation functions and tied weights. The input layer has 81 neurons which equals the total number of features in the CICIDS2017 dataset. The first hidden layer of the sparse auto-encoder was able to successfully reduce the dimensions to 70 features with a good error approximation. Further, the features were reduced to 64 in the second hidden layer. Once the weights are trained, the resulting sparse auto-encoder can be used to perform the classification in the final stage. The parameters of the sparse representation are set as follows: the weight decay $\lambda = 0.0008$. The weights are multiplied by λ to prevent the weights from growing too large. The sparsity parameter $\rho = 0.05$, and the sparsity penalty term $\beta = 6$. The sparsity parameters and penalty are designed to restrict the activation of the hidden units, which reduces the dependency between the features. The algorithm is summarized in Table 3.1 and the design principles are presented in Table 3.2.

3.3.2 Principal Component Analysis Analysis (PCA) Based Feature Reduction

In this section, we present the Principal Component Analysis (PCA) algorithm. The objective of PCA is to perform dimensionality reduction. PCA finds a transformation that reduces the dimensionality of the data while accounting for as much variance as possible. PCA is the oldest technique in multivariate analysis. The fundamental concept of the PCA is the projection-based mechanism. Here, the original dataset $X \in R^n$ with n columns (features) is projected into a subspace with k or lower dimensions representation $X \in R^K$ (fewer columns), while retaining the essence of the original data. The algorithm works as follows:

Table 3.1: Pseudo-code for the proposed Auto-Encoder.

Dimensionality Reduction Using AE
Training:
1. Perform the feedforward pass on all the training instances and compute
$a^{(1)}, a^{(2)}$ Equation (3.1)
2. Compute the output, sparsity mean, and the error of the cost function
$J(W, b; \hat{x}, x)$ Equation (3.2)
$\hat{\rho}_j$ Equation (3.3)
3. Compute the cost function of the sparse auto-encoder
$J_{sparse}(W, b)$ Equation (3.4)
4. Backpropagate the error to update the weights and the bias for all the layers
Dimensionality Reduction:
Compute the reduced features from the hidden layer
$a^{(1)}$ Equation (3.1)

Table 3.2: Design Principles

Parameters	**Value**	**Description**
λ	0.0008	Weight decay
β	6	Sparsity penalty
ρ	0.05	Sparsity parameter

To reduce the features dimensionality from n-dimensions to k-dimensions, two phases are implemented; the preprocessing phase and the dimensionality reduction phase. In the preprocessing phase, (steps 1 through 4 below), the data is preprocessed to normalize its mean and variance using Equations (3.6) and (3.7). In the second phase (steps 5 through 8), which represent the reduction phase, the covariance matrix Cov_M, Eigen-vectors and Eigen-values are calculated from Equations (3.8) and (3.9).

(1) Normalize the the original feature values of data by its mean and variance using Equation (3.6), where m is the number of instances in the dataset and

$X_{(i)}$ are the data points.

$$\mu = \frac{1}{m}\sum_{i=1}^{m} X_{(i)} \tag{3.6}$$

(2) Replace $X_{(i)}$ with $X_{(i)} - \mu$.

(3) Rescale each vector $X_{j(i)}$ to have unit variance using Equation (3.7).

$$\sigma_j^2 = \frac{1}{m}\sum_{i} (X_{j(i)})^2 \tag{3.7}$$

(4) Replace each $X_{j(i)}$ with $\frac{X_{j(i)}}{\sigma}$.

(5) Compute the Covariance Matrix Cov_M as follows:

$$Cov_M = \frac{1}{m}\sum (X_{(i)})(X_{(i)})^\top \tag{3.8}$$

(6) Calculate the Eigen-vectors and corresponding Eigen-values of Cov_M.

(7) Sort the Eigen-vectors by decreasing the Eigen-values and choose k Eigen-vectors with the largest Eigen-values to form W.

(8) Use W to transform the samples onto the new subspace using Equation (3.9).

$$y = W^T \times X \tag{3.9}$$

where X is a $d \times 1$ dimensional vector representing one sample, and y is the transformed $k \times 1$ dimensional sample in the new subspace.

The computational complexity of executing the designed PCA depends on the number of features P that represent each data point [174].

$$O(P^3) \tag{3.10}$$

According to [175], the Reduction Ratio (RR) of PCA can be defined as the ratio of the number of target dimensions to the number of original dimensions. The lower the value of RR, the higher is the efficiency of PCA.

3.4 Classification

There are various types of classifier models that have been developed and used for different types of applications and in different fields of science. Each of these algorithms has its unique characteristics while others share the same concept. The process of choosing a certain type of classifier model may affect the final accuracy result, time to build the model, and detection rate. Figure 3.4 summarizes previous studies that used AWID to design an IDS with different classification algorithms. From this Figure, we notice that approximately 17% of the IDS methods use classification schemes that are based on Artificial Neural Network (ANN) classifiers. Next are Random Forest, J48, OneR, ZeroR, and AdaBoost (11% for each classifier), 6% based on Voting and Extra Tree, and 5% based on Random Tree. Finally, approximately, 5% are non-classification (associative rule) based. In the remainder of this section we highlight the eleven classifiers that we used in the classification process.

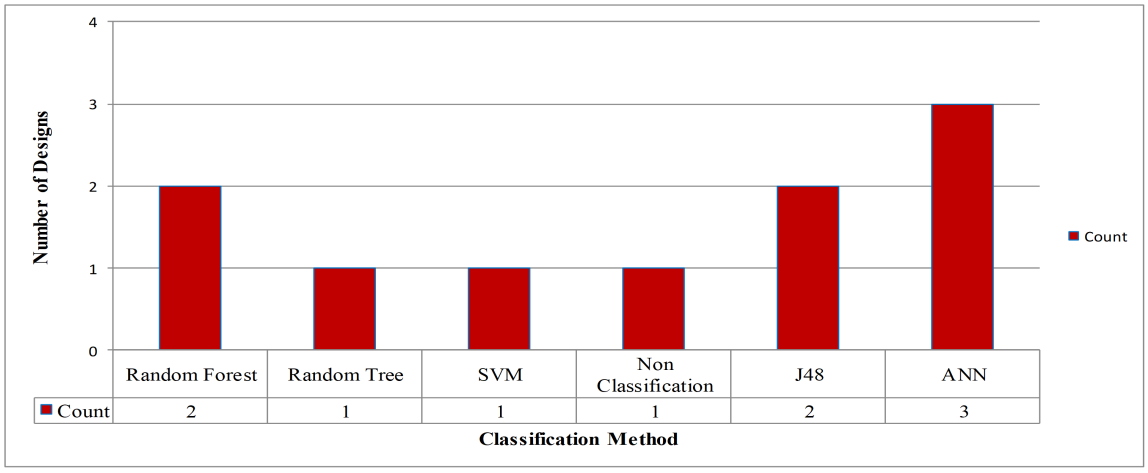

Figure 3.4: Statistics of relevant ML classification methods in AWID

3.4.1 Artificial Neural Network (ANN)

ANN is a learning algorithm built for information processing through mathematical or computational models. There are different types of ANNs, such as Multilayer Perceptron (MLP), and Learning Vector Quantization (LVQ). However, a neural network (NN) is typically a number of interconnected neurons in which each connection has a weight associated with it. During the classification process, it can predict the correct class label of the input layer by modifying the weights. In general, the NN consists of an input layer, one or more hidden layers and an output layer. The choice of selecting the number of hidden layers is different and depends on each application. In practice, one layer is usually used [176]. Generally, the architecture of ANN uses different layers.

3.4.2 Naïve Bayes (NB)

Naïve Bayes (NB) is a statistical classifier based on the Bayes theorem with robust probabilities and assumptions. It is a simple probabilistic based machine learning classifier that was introduced in 1960. It incorporates Bayes theorem to perform classification, and can predict class probabilities by a given tuple belonging to a specific class. The assumption of a Bayesian classifier is called class conditional independence, which means each feature value of a specific class has independent effects from other feature values. Theoretically,it is a superior classifier that can guarantee high classification speed and accuracy when applied to large databases [177].

3.4.3 Random Forest (RF)

Random Forest, introduced by Leo Breiman in 2001 [178, 179], is a multiple decision tree based algorithm. Random forest creates many classification trees to

classify new inputs. Hence, it applies the input vector to each of the trees in the forest and then selects a class that most of trees produced [179]. Random Forest algorithm runs efficiently on large data sets. Furthermore, Random Forest can carry many input variables as well as it efficiently handles large percentage of missing data while maintaining accuracy [180]. The computing time required to establish the RF classification model is [181, 182]:

$$T \times \sqrt{MN \log(N)} \tag{3.11}$$

where T is the number of trees, M is the number of variables used in each split, and N is the number of training samples.

3.4.4 Bagging

Bagging is a machine learning ensemble meta-algorithm [183]. This type of algorithm is designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. The Bagging (bootstrap aggregating) is a meta-algorithm. This technique reduces the variance while retaining the bias, which improves the accuracy of a single model through using multiple copies of it trained on different sets of data. The bagging technique can overcome over-fitting by reducing prediction variance.

3.4.5 AdaBoost

The AdaBoost algorithm, introduced by Freund and Schapire, is the first and the most commonly used boosting algorithm [184]. AdaBoost is being used in applications of numerous fields such as intrusion detection systems [23, 24]. Through boosting, AdaBoost improves the performance by combining many relatively weak and inaccurate rules generated by other learning algorithms. These other learning algorithms are referred to as weak learners or base learner components. The weak

learner is used to generate a hypothesis for each sample in the training dataset. AdaBoost combines weak hypotheses generated by weak learners in many rounds to generate improved hypothesis [23].

3.4.6 Simple Logistic

This classifier is used to build a linear logistic regression model. Here, boosting classifier with simple regression functions are used as base learners to fit the logistic models [185].

3.4.7 LogitBoost

LogitBoost is a boosting classification algorithm. LogitBoost and AdaBoost are close to each other in the sense that both perform an additive logistic regression. The difference is that AdaBoost minimizes the exponential loss, whereas LogitBoost minimizes the logistic loss [184].

3.4.8 OneR

OneR is a rule based classification algorithm. It is a very simple classification algorithm based on a single rule; constructing a frequency table for each predictor against the target, for each predictor in the data. It then selects the rule with the smallest total error as its "one ruleâ. OneR ranks features (attributes) of the training dataset using error rate and selects the most informative attribute to develop a rule that predicts the class of the data [186]. The OneR algorithm requires discrete attributes. If not, they are discretized.

3.4.9 Random Tree

The Random Tree algorithm is a tree-based classification algorithm. It employs a single tree for data classification and it uses randomly selected number of attributes (say K) at each node of the tree for the classification. The random forest classifier is an ensemble method usually applied to Random Tree.

3.4.10 ZeroR

ZeroR is the simplest classification method which relies on the target and ignores all the predictors. The ZeroR classifier simply predicts the majority category (class). ZeroR can be used to determine the baseline performance as a baseline classifier for other classification methods [187]. Both ZeroR and OneR classifiers consider a frequency table to predict a class. The difference is that OneR depends on both the frequency table and one constructed rule from the frequency table.

3.4.11 J48

J48 is a Decision Tree (DT) based classification algorithm introduced by Ross Quinlan. DT is a nonlinear classifier that has been mainly used for solving problems related to machine learning and classifier systems [188]. The learning and classification steps of such algorithms are simple, fast and can handle multi-dimensional data. In addition, it can work with noisy data and missing data in a database [189]. The concept of DT classifiers is a flowchart-like tree structure in which each decision discards a certain class until reaching the accepted class. It consists of nodes, branches, and leaves. A node denotes a test on an attribute using transition rules, a branch represents the result of the test, and the leaf holds a class label.

3.4.12 Linear Discrimination Analysis (LDA)

Linear Discriminant Analysis (LDA) is a method used in machine learning to find a linear combination of features that characterizes or separates two or more classes of instances. The resulting combination may be used as a linear classifier. LDA is also closely related to principal component analysis (PCA) and factor analysis in that they both look for linear combinations of variables which best explain the data [168]. LDA explicitly attempts to model the difference between the classes of data. The LDA is a type of Bayesian classifier. The LDA assumes the same $\sum$ for all classes. Essentially, LDA computes a separate μ for each class (using training points that belong to it), but $\sum$ is computed using the entire training data. This common covariance matrix is used in the computations corresponding to every class.

3.4.13 Quadratic Discriminant Analysis (QDA)

A Quadratic classifier is used in machine learning to separate measurements of two or more classes of instances by a quadratic surface. QDA is a type of Bayesian classifier. QDA computes a separate $\sum$ and μ for each possible output class.

3.5 Imbalanced Class Handling Approaches

The problem of learning from skewed multi-class datasets is an important topic that arises very often in practice in classification problems [190]. In such problems, almost all the instances are labelled as one class, while far fewer instances are labeled as the other class or other classes; usually the more important class. According to [191], the Imbalance Ratio (IR) can be defined as the ratio of the number of instances in the majority class to the number of instances in the minority class, as presented

in Equation 3.12.

$$Imbalance\ Ratio = \frac{Majority\ Class\ Instances}{Minority\ Class\ Instances} \tag{3.12}$$

Nowadays, imbalanced class distribution is a major hurdle. If the IR value in the data is high, classifiers will be lower in accuracy and reliability; i.e. they do not truly reflect the classes accurately. Furthermore, imbalanced class distribution is an inevitable problem in real traffic due to the large size of traffic and low frequency of certain types of anomalies.

The following section, provides brief description of the common approaches for handling imbalanced datasets [190], which include : re-sample with replacement (RWR), re-sample without replacement (RWoR), SMOTE (Synthetic Minority Over-sampling), and Class Balancing (or class balancer).

3.5.1 Balancing Approaches at Data Level

The aim of these approaches is to balance and normalize the class distribution before passing the dataset to the classifier. Sampling is the most commonly used approach for overcoming miss-classification problems due to imbalanced datasets. The basic approaches at data level are explored further next:

3.5.1.1 Re-Sample Without Replacement (RWoR)

Re-Sample Without Replacement is a non-heuristic method that aims to balance class distribution through the random elimination of majority class samples. The rationale behind it is to try to balance out the dataset in an attempt to overcome the idiosyncrasies of the machine learning algorithm. This technique produces a random sub-sample of a dataset [192].

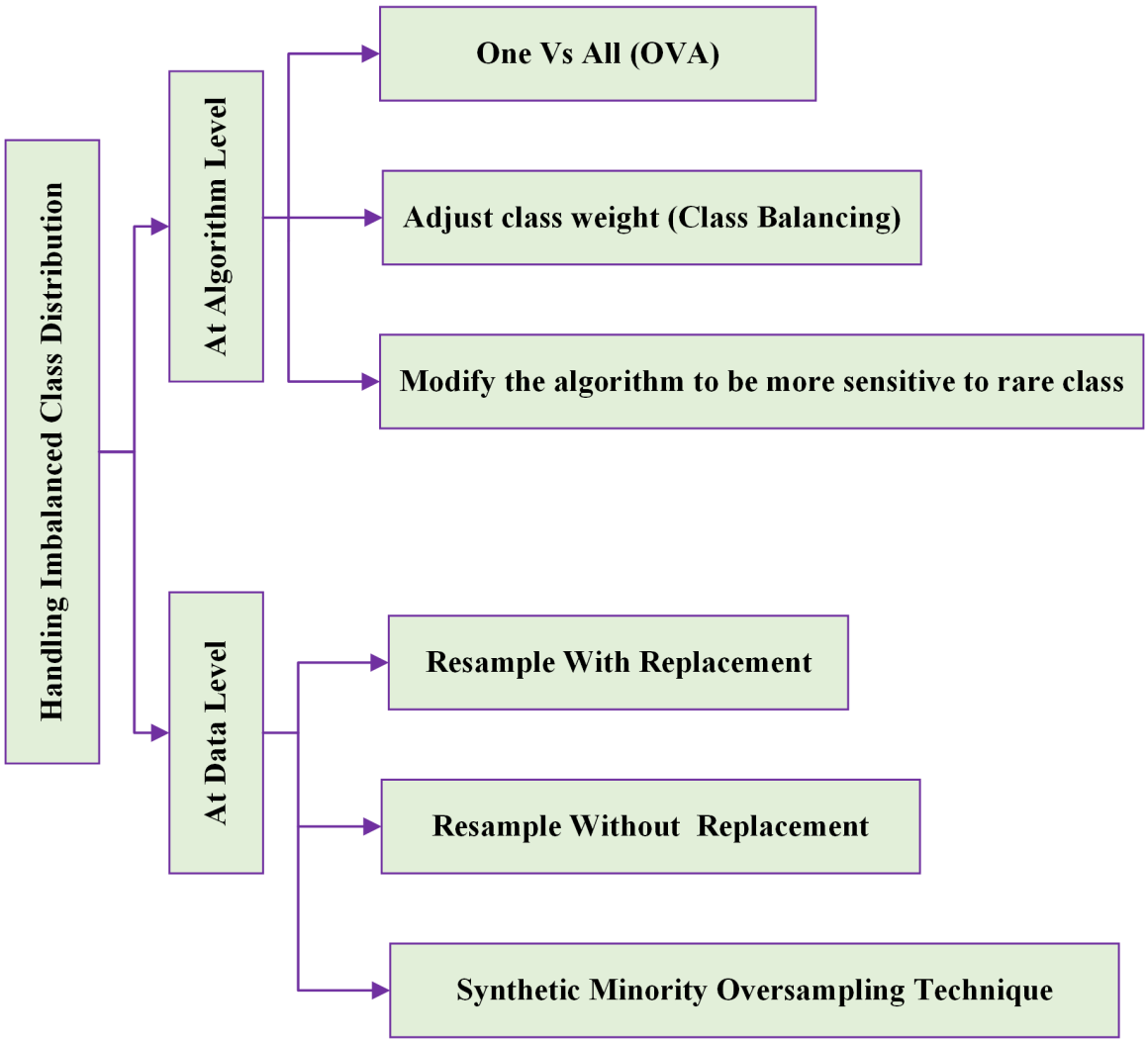

Figure 3.5: A taxonomy of imbalanced class handling approaches

3.5.1.2 Re-Sample With Replacement (RWR)

Random over-sampling is a non-heuristic method that aims to balance class distribution through the random replication of minority class samples [193]. This approach can increase the likelihood of occurring over-fitting [193], since it makes exact copies of the minority class samples. This way, a symbolic classifier, for instance, might construct rules that are apparently accurate, but actually cover one replicated example. In addition, oversampling can introduce an additional computational task

if the dataset is already fairly large, but imbalanced.

3.5.1.3 Synthetic Minority Oversampling Technique (SMOTE)

This approach generates synthetic minority examples (samples) to over-sample the minority class [194]. Its main idea is to form new minority class examples by interpolating between several minority class examples that lie together. For every minority example, its k (which is set to 5 in SMOTE) nearest neighbors of the same class are calculated, then some examples are randomly selected from them according to the over-sampling rate. After that, new synthetic examples are generated along the line between the minority example and its selected nearest neighbors. Thus, the over-fitting problem is avoided and causes the decision boundaries for the minority class to spread further into the majority class.

3.5.2 Distribution Based Balancing (DBB)

This section provides a glance at the distribution based balancing (DBB) technique [195]. DBB is based on learning and sampling probability distributions. In this technique, the sampling of instances is performed following a distribution learned for each pair example of feature and class label. More specifically, the user determines the distribution balancing type to learn in order to re-sample new instances. According to [191], the Imbalance Ratio (IR) can be defined as the ratio of the number of instances in the majority class to the number of instances in the minority class, as presented in Equation 3.12.

For the CICIDS2017 dataset, IR is 5:1 and the total number of classes is 15 classes. To apply Uniform Distribution Based Balancing (UDBB) [196], a uniform number of instances ($I_{Resample}$) for each class is calculated from Equation 3.13. The

pseudo-code is presented in Table 3.3.

$$I_{Resample} = \frac{Number\ of\ Instances\ in\ the\ dataset}{Number\ of\ Classes\ in\ the\ dataset} \tag{3.13}$$

Table 3.3: UDBB pseudo-code

Input Training Set: D_{Train}
Set Distribution to Uniform
C : Number of Classes
F_T : Total number of features in D_{Train} Training Set
I_{old} : Total number of Instances in D_{Train}
Calculate the required number of Instances in each class: $I_{Resample}$
Training Set $D_{Train_{new}} = \emptyset$
For each class C_i Do
While $i \neq I_{Resample}$
For each feature $F_1, ..., F_T$
Generate new sample using uniform distribution
Assign Class label
Return $D_{Train_{new}}$

3.5.3 Balancing Approaches at Algorithm Level

The balancing approaches at algorithm level include adjusting the costs of the various classes so as to counter the imbalanced classes, adjusting the class weight, adjusting probabilistic estimate at the tree leaf, adjusting the decision threshold, and one vs. all. This dissertation applies the adjusting class weight and the one vs. all approaches.

3.5.3.1 Adjust Class Weight: Class Balancing (CB)

This approach re-weights the instances in the data so that each class has the same total weight. The total sum of weights across all instances will be maintained.

3.5.3.2 One-Vs-All (OVA)

OVA decomposition divides an m class problem into m binary problems. Each problem is faced by a binary classifier, which is responsible for distinguishing one of the classes from all other classes. The learning step of the classifiers is done using the whole training data, considering the patterns from the single class as positives and all other examples as negatives [15].

3.6 IDS for Cyber Physical System

Behavior-based detection approaches can benefit CPS because these systems can deliver a well-defined concept of cognitive processes, which can exploit its consistent behavior. The expectant detection techniques must have the ability to identify real attacks from random defects, ingrained defects in the design, misconfiguration of the system devices, system faults, human errors, and software implementation bugs. Therefore, state estimation is an effective method to adopt for CPS's.

3.6.1 Hardware Based Behaviour Rules IDS

A medical device is defined as an instrument used alone or in combination with other instruments to monitor and treat human beings for one or more of specific medical conditions [197, 140]. Different device models are networkable and can send their output to a central monitoring station, from which healthcare personal process the medical data simultaneously. Medical devices are often characterized and controlled by sophisticated patient treatment algorithms that interact with the physical environment and the patient [149]. The most prominent characteristic of a medical device is its feedback loop that reacts to the physical environment. For example, in portable, wearable and transportable devices such as Continuous Glucose Monitor

Table 3.4: Malicious behavior rule in CNF

Device	Malicious State	States Components
PCAg	(Analgesic Request = TRUE)Λ(Pulse <T)	Analgesic Request, Pulse
PCAg	(Analgesic Request = TRUE)Λ(Respiration <T)	Analgesic request, Respiration
CD	(Analgesic Infusion Rate >0) Λ (Mode=Defibrillator)	Analgesic Infusion Rate and CD Mode
CD	(Mode =PACEMAKER)Λ ($\|Pulse - PacemakerFrequency\|$)$\delta_F$	CD Mode, Pulse
VSM	$\|Monitor\ Temperature - Trustee\ Temperature\| > \delta_T$	Temperature
VSM	$\|Monitor\ Respiration - Trustee\ Respiration\| > \delta_R$	Respiration
CGM	Insulin Request Rate >T	Insulin Request Rate
CGM	(Insulin Request = TRUE) Λ (Pulse <T) Λ (Glucose <T)	Insulin Request Rate, Pulse, Glucose

(CGM), the actual device transmits the patient's data via a wireless data connection. Other devices such as Vital Sign Monitor (VSM) and Patient Control Analgesia (PCAg) which are non-portable, non- wearable and non-transportable, can send their output using wireline communication to a central monitoring station inside a hospital. This is while other medical devices such as Cardiac Devices (CD) are implanted in the patient and do not use wireline communication. In this section, four types of medical devices are studied. This dissertation introduces a hardware approach implementing lightweight IDS monitoring tools for VSM, PCAg, CD, and CGM devices that can be mapped onto an FPGA chip which can then be embedded into the medical device. Furthermore, this research proposes behavior rules for a CGM device, and transforms these behavior rules to state machines to build the hardware module. The hardware module uses the set of behavior rules along with the related readings and settings of sensors and actuators to detect if a device's behavior deviates from the expected normal behavior. It provides an output to distinguish among system states that are safe, unsafe, warning, and idle. Tables 3.5 and 3.4 present the set of normal behavior rules and malicious behavior rules in the Conjunctive Normal Form (CNF) representation.

Table 3.5: Normal behavior rule in CNF

Description	Safe State	Trustee	Monitor
Pulse above threshold during analgesic request	(Analgesic Request = TRUE) Λ(Pulse $>T_s$)	PCAg	VSM
Analgesic request rate below safe threshold	Analgesic Request Rate $>T_s$	PCAg	VSM
Pulse matches pacemaker frequency	Pulse = Pacemaker Frequency	CD	VSM
Patient is unstable before defibrillation	(Pulse $<T_s$) Λ (Respiration $<T_s$) Λ (CD Mode = Pacemaker)	CD	VSM
Trustee blood pressure matches monitor	Trustee blood pressure = Monitor blood pressure	VSM	Peer VSM
Trustee oxygen saturation matches monitor	Trustee oxygen saturation =Monitor oxygen saturation	VSM	Peer VSM
Glucose above threshold during insulin request	(Insulin Request = TRUE) Λ (Glucose $>T_s$)	VSM	Peer VSM
Insulin request rate below safe threshold	(Insulin Request Rate $<T_s$)	GSM	VSM

3.6.2 Transforming Behavior Rules to State Machines

In the process of transforming behavior rules to state machines, the "malicious behavior state" is identified as a result of a behavior rule being violated [140] [198]. Thus, the initial step is to identify malicious behavior for a specific device as a state. This state is then converted into the Conjunctive Normal Form (CNF) predicate and identifies the involved state components in the underlying state machine. Following that, a Boolean expression in the Disjunctive Normal Form (DNF) is used to group together all the extracted malicious behavior states that are related to each device, as shown in the following DNF:

CGM. (Insulin Request Rate $>T$) $\vee$ ((Insulin Request = TRUE) $\wedge$ (Pulse $<$ T) $\wedge$ (Glucose $<T$)) $\vee$ ((Insulin Request = TRUE) $\wedge$ (Respiration $<T$) $\wedge$ (Glucose $<T$)) $\wedge$ ($|Monitor\ Glucose - Trustee\ Glucose| > \delta$)

A disjunctive normal form (DNF) can be defined as a normalization of a logical formula. It can also be described as an OR of ANDs, a sum ($\vee$) of products ($\wedge$), or (in philosophical logic) a cluster concept [199]. Next, the union of all the predicate variables is converted into state components of a state machine to establish their corresponding ranges. Finally, the managing process will collapse and identify the sequence of values that are not legitimate in order to reduce the total number of states in the states' space. Table 3.6 indicates the symbols used in this study.

Table 3.6: Symbols used

Symbol	Parameters	Note
T	Threshold value	Each value is tied to a specific device system state (warning, unsafe), and certain state component
δ	Sensor Reading Deviation	Each value is tied to a specific device specific patient, and specific system states (warning, unsafe, safe)
L	Acceptable Low Heart Rate Pulse	Each value is tied to a specific device specific system state, and patient
H	Acceptable Low Heart Rate Pulse	Each value is tied to a specific device specific system state, and patient
T_s	Safe threshold value	This value is tried to a specific device specific patient, and specific system state (safe)

CHAPTER 4: IMPLEMENTATION AND EVALUATION

This chapter outlines the plan for: i) implementing and testing the proposed intrusion detection framework; mainly the software-based machine learning ides and the expected outcomes and objectives of the this approach. The work is implemented using the WEKA 3.9 and Python programming tools; and ii) implementing and testing the medical CPS IDS.

4.1 Software Machine Learning Based IDS

In this section, we present our experimental study plan and procedure. The feature selection and reduction ideas, as well as different balancing techniques and various classifiers have been applied on different benchmark IDS datasets.

4.1.1 AWID

As shown in Figure 4.1, our procedure starts with locating and eliminating redundant and unnecessary features in AWID-ATK-R. Once the features were chosen, it was necessary to perform preprocessing, followed by normalization. In the follow-up steps of the experimental study, different methods were used to select the optimal feature sets. To obtain the sets, several known machine learning classifiers were trained to identify the most efficient ones. The final stage of the study included testing

the trained classifier models. In the following subsections, each step is presented in more detail.

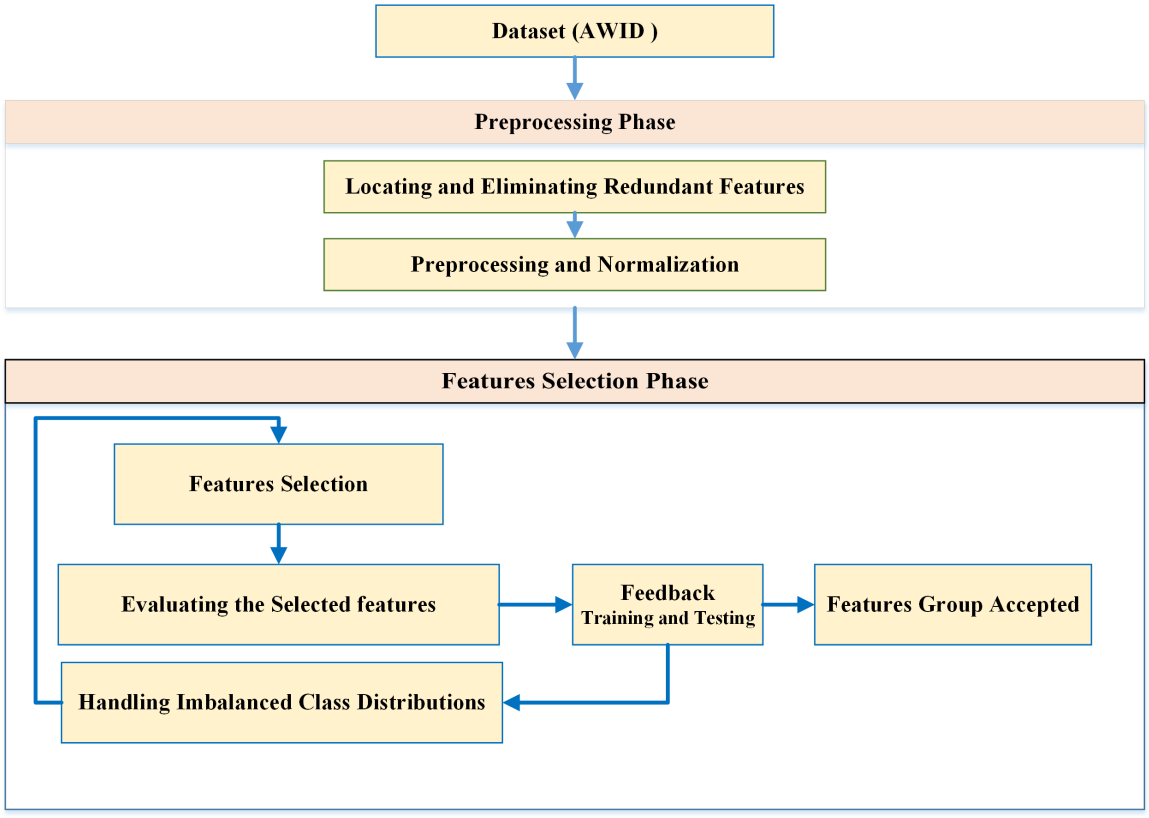

Figure 4.1: Proposed intrusion detection based on the AWID dataset

The AWID-ATK-R-Trn dataset was used to train the machine learning techniques and the AWID-ATK-R-Tst dataset was used to evaluate the performance of the machine learning techniques.

4.1.1.1 Preprocessing

Preprocessing is a vital component in the machine learning experiments and plays a key role in the classifier model as well as enhancing the overall classification performance. To begin this process, we determine the feature types of the dataset which include numerical and non-numerical data. To be more specific, the service set

identifier (SSID) feature is a non-numerical feature. It is a string. Other features are numeric. This work has traditionally employed a simple preprocessing stage were the hexadecimal values in AWID are transformed into integer representation. After the preprocessing step, the features' scales within the dataset were heavily imbalanced. Thus, normalization was applied on the selected sets.

4.1.1.2 Unity-Based Normalization

In this section, we use Equation 4.1 to re-scale the features in the dataset based on the minimum and the maximum values of each feature. Some features in the dataset vary between [0, 1] while other features vary between [0, ∞). Therefore, these features were normalized to restrict the range of the values between 0 and 1.

$$x_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

where x_i is the value of a particular feature, x_{min} is the minimum value, and x_{max} is the maximum value.

4.1.1.3 Feature Selection Procedures

Some of the features are vital for intrusion detection, whereas other features may act as noise; causing a negative impact on the training speed and the accuracy. Therefore, in the first step of our experiments, a total of 32 attributes were selected based upon reliable and accurate manual attribute selection and recommendation from previous related work [91, 200]. Cited work [201] ranked the most important attributes as those related to the MAC header. In addition, in previous studies of wireless intrusion detection systems (WIDS), certain attributes have been found to be related to the intrusion detection process [2, 89, 98, 202]. Table 4.1 shows the 32 attributes group set (32-AGS), also noted as 32 features set group (32-FSG).

In the second step, we used the Correlation Feature Selection (CFS) mea-

Table 4.1: AWID 32-FSG

Feature Name	Feature Name
epoch Time	wlan.fc.pwrmgt
frame.time_delta	wlan.fc.protected
frame.time.relative	wlan.duration
frme.len	wlan.ra
frame.ignored	wlan.da
radiotap.mactime	wlan.ta
radiotap.channel.type.cck	wlan.sa
radiotap.dbm_antsignal	wlan.bssid
wlan.fc.type_subtype	wlan.seq
wlan.fc.type	wlan_mgt.fixed.listen_ival
wlan.fc.subtype	wlan_mgt.fixed.timestamp
wlan.fc.ds	wlan_mgt.fixed.beacon
wlan.fc.retry	wlan_mgt.fixed.reason_code
wlan_mgt.tim.dtim_period	wlan_mgt.tagged.all
wlan.wep.iv	wlan.wep.key
wlan.wep.icv	data.len

sure [25] to evaluate the attributes space along with the Best First Search method of forward direction [160], which is a heuristic search strategy to evaluate the attributes and select the highly class-correlated ones, yet uncorrelated to one another. Based on that, 10 attributes were selected. These attributes are listed as follows: frame.time.relative, frame.len, radiotap.channel.type.cck, wlan.fc.subtype, wlan.fc.dc, wlan.fc.pwrmgt, wlan.ta, wlan.seq, wlan.wep.iv, and data.len.

In the third step, we used the Harmony Search (HS) algorithm [152] which is one of the meta-heuristic algorithms that mimics the improvisation process of music players. One advantage of HS is that it avoids the problem of imposing complicated mathematical requirements and it is not sensitive to the initial value settings [152]. Here, we incorporate the HS algorithm with the Cost Sensitive Subset Evaluator (CostSensitiveSubsetEval), which is a meta subset evaluator that makes its base subset evaluator cost-sensitive [203]. The algorithm takes a cost matrix and a

base evaluator. If the base evaluator can handle instance weights, then the training data is weighted according to the cost matrix, otherwise the training data is sampled according to the cost matrix. Based on this, a 7 Attributes Group Set (7-AGS) was selected that is consisted of radiotap.mactime, radiotap.channel.type.cck, wlan.fc.subtype, wlan.fc.pwrmgt, wlan.bssid, wlan.seq, and data.len.

Finally, we used the CFS algorithm along with the Harmony Search technique [152], based upon which, the 5 Attributes Group Set (5-AGS) was selected. The list of this 5-AGS are epoch.time, frame.len, wlan.duration, wlan.ra, and data.len.

4.1.2 CICIDS2017

We utilized the up-to-date CICIDS2017 intrusion detection and prevention dataset [8], which consists of five separated data files. Each file represents the network traffic flow and specific types of attacks for a certain period of time. To be more specific, the dataset was collected based on a total of 5 days, Monday through Friday. Monday is the normal day and only includes the benign network traffic, whereas the implemented attacks in the dataset were executed on Tuesday, Wednesday, Thursday and Friday. This dissertation combined all CICIDS2017's files together and fed them through the AE and PCA units for a compressed and lower dimensional representation of all the fused data. Figure 4.2 displays the idea of the proposed framework.

4.1.2.1 Preprocessing

In this study, a preprocessing function is applied to the CICIDS2017 dataset by mapping the IP (Internet Protocol) address to an integer representation. The mapped IP includes the Source IP Address (Src IP) as well as the Destination IP Address (Dst IP). These two are converted to an integer number representation. This

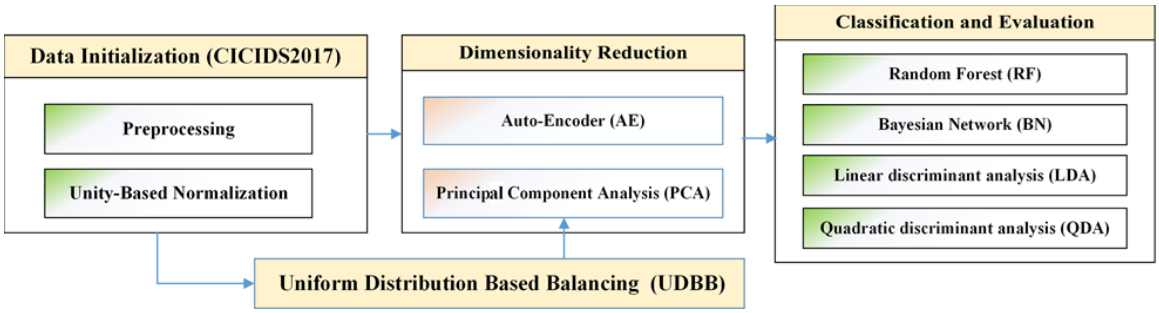

Figure 4.2: Anomaly-based intrusions detection on CICIDS2017

study splits the data into training set and testing set with a ratio of 70:30.

4.1.2.2 Unity-Based Normalization

In this step, we use Equation 4.1 to re-scale the features in the dataset based on the minimum and maximum values of each feature. Some features in the original dataset vary between [0, 1] while other features vary between [0, ∞). Therefore, these features are normalized to restrict the range of the values between 0 and 1, which are then processed by the auto-encoder for feature reduction.

4.1.2.3 Features Reduction Procedures

This dissertation accustoms Auto-Encoder (AE) [88],[196],[204],[205],[206],[207] and Principal Component Analysis (PCA) for dimensionality reduction [196],[208],[205] and [164],[209]. Then, these reduced features are utilized to develop of a framework for netflow based intrusion detection.

4.1.3 CIDDS-001

This section outlines the essential steps of our methodology and experimental procedures on the CIDDS-001 dataset. See Figure 4.3.

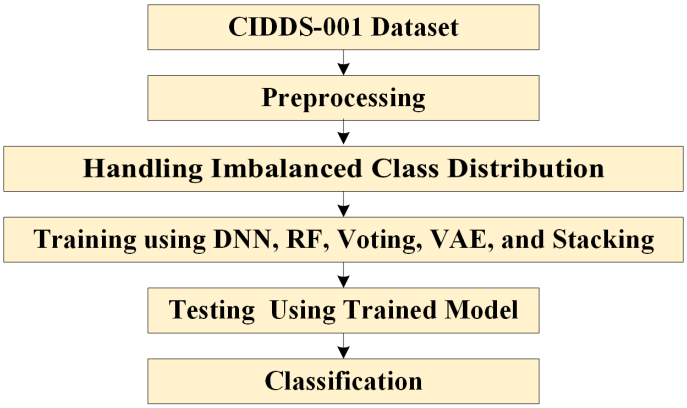

Figure 4.3: Experimental procedure applied on the CIDDS-001 intrusion dataset

4.1.3.1 Preprocessing

In this experiment, a simple preprocessing stage was deployed where the hexadecimal values in the dataset are transformed into integer representation. Since after this step, the features' scales within the dataset were heavily imbalanced, normalization was applied on the selected sets to bring the ranges between 1 and -1.

4.1.3.2 Experimental Procedures

This section explains in detail the experimental procedure. This research utilizes the following machine learning classification approaches for intrusion detection in binary classification on the CIDDS-001 benchmark dataset:

- Deep Neural Networks (DNN) with adaptive learning rates and architectures as tabulated in Table 4.2
- Random Forest (RF)
- Voting technique to vote on OneR, Naïve Bayes, ExtraTree
- Stacking technique with Linear discriminant analysis (LDA), Naïve Bayes, and OneR

Table 4.2: Proposed DNN architecture

Classifier	#of Layers	#of Neurons in each layer	Learning Rate
DNN (3,512,0.1)	3	512	0.1
DNN (3,1024,0.1)	3	1024	0.1
DNN (3,2048,0.1)	3	2048	0.1

Moreover, we carried out our experiments using two scenarios: With Handling Imbalanced Class Distribution (WHICD), and Without Handling Imbalanced Class Distribution (WoHICD). In the WHICD scenario, we apply sampling techniques to handle class distribution of the CIDDS-001 before feeding the data to the classifiers. In the WoHICD scenario, we apply the machine learning classifiers to the original class distribution. We compare and contrast the two scenarios for the classifiers. The techniques [210] that we employed on CIDDS-001 to handle imbalanced class distributions include the following:

- Up-sampling the minority class
- Down-sampling the majority class
- Spread Sub-Sample
- Class Balancer

Up-sampling [211] is defined as the procedure of randomly duplicating instances from the minority class in order to reinforce its signal. This is while Down-sampling [211] involves randomly removing observations from the majority class to prevent its signal from dominating the learning algorithm. The Spread Sub-Sample technique produces a random sub-sample of the dataset. The technique allows to specify the Distribution Spread, which is the the maximum class distribution spread that represents the maximum "spread" between the minority and majority classes. To illustrate, it may specify that there be at most a 2:1 difference in class frequencies. In our experiments,

we set the distribution spread to a uniform distribution (Distribution Spread=1). Finally, the Class Balancer re-weights the instances in the data so that each class has the same total weight. Moreover, the total sum of weights across all instances will be maintained. In our experimental procedure, we keep the original distribution for the test dataset as is without applying any of the handling imbalanced class techniques since the status of the network flow traffic tends to have skewed towards normal traffic. Table 4.3 shows the CIDDS-001 class distribution before and after the imbalanced class handling (balancing) approaches [92].

Table 4.3: CIDDS-001 class distribution

Handling Imbalanced Class Approach	No. of Instances	
	Attack	Normal
Spread Sub-Sample	58,360	58,360
Down-Sample the Majority Class	58,359	1,508,500
Up-Sample the Minority Class	1,508,499	1,508,500
Original Distribution	58,359	2,958,640

4.2 Hardware Behavior Rules Based Co-Design

Four medical devices were employed in our research for the MCPS case study [140]. The proposed Behavior Specification Rule Monitoring (BSRM) tool is based on a set of behavior rules that have been determined and designated during the debugging and operational phases to specify acceptable behaviors of sensors and actuators embedded in medical devices such as VSM, PCAg, CD and CGM. Accordingly, BSRM will identify whether each medical device's behavior is normal or malicious according to the set of behavior rules. The observations are composed from the audit data that were represented by the logs generated by the relevant sensor or actuator drivers, the MIMIC-III [212] Dataset, and the University of Queensland vital signs Dataset [213]. The test bench incorporates different acceptable parameter ranges that reflect the

physiology and responses for patient treatment related to each device's state components. BSRM operates through individual medical devices, since BSRM has no designated monitoring node, there is no single point of failure. In a VSM example, one VSM is monitored by another peer VSM for security purposes. In a PCAg, one PCAg is being monitored by a VSM and a peer PCAg.

4.3 State Transition Diagram

The normal behavior states as well as the malicious behavior states are used to build a state transition diagram for each specified device's behavior monitoring tool. The state components related to the designated state machine of our model are shown in Table 4.5. We use these state components along with both normal and malicious behavior rules, sensors and actuators readings, and settings to build an idealized model of a finite state machine engine in Hardware Description Language (HDL) for monitoring each device's behavior. Each designed state diagram consists of four states that include: Idle, Safe, Unsafe, and Warning state. Table 4.6 indicates the number of the device states and the reasons why the device entered such states. It is worth to mention that the unsafe state of the state machines are not those "hazardous" states generated due to design faults (e.g. software bugs). Such "hazardous" states are removed as a result of the design faults that were identified during the testing and debugging phases. In our research, the safe states and unsafe states are permanent and are based on either malicious or normal behavior in a specific medical device. The idle state represents the initial status of the system module. Through an asynchronous reset signal, the system can be forced to enter this state. During system initialization, this state is added to the Finite State Machine (FSM) of the system. Furthermore, the system is in a warning state as long as its specified parameters exceed the warning threshold for at least one behavior rule. This state may indicate an earlier sign of

malicious behavior or represents a slight difference in readings of the two related sensors due to environmental influences such as indoor noise. Moreover, a safe state represents normal behavior of the specified device and indicates that the system follows its designated normal behavior rules. On the contrary, an unsafe state reflects malicious behavior. Figure 4.4 shows a state transition diagram of a device's behavior pattern.

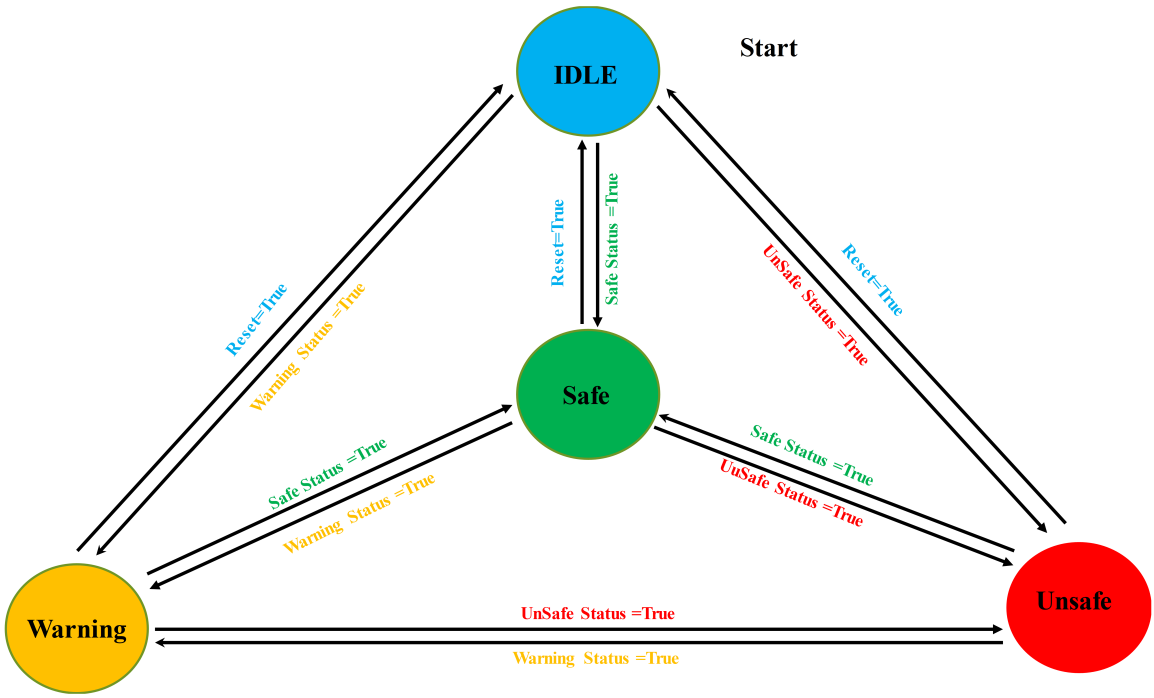

Figure 4.4: State machine diagram

The following explanations have been added to illustrate and describe the malicious behaviors in the medical devices under analysis.

One state could be when the PCAg device acts as a trustee, while the VSM device acts as a monitor. Here, when the PCAg device receives the patient's request for analgesic to relieve pain, the PCAg device will distribute the necessary analgesic according to the patient's request. Once the patient receives pain relief, the patient's vital signs displayed on the VSM device would return to a relaxed state. One can infer that all

of the patient's vital signs are stable. However, if the PCAg device is compromised through malicious behavior, then the PCAg device will continuously administer the analgesic, and the final result would be an overdosed patient. Therefore, if the PCAg continues to receive additional requests for analgesic while a patient's pulse or respiration rate is below a normal threshold, then malicious behavior is present in this device. Furthermore, if the analgesic request rate of the PCAg device exceeds normal threshold, then malicious behavior is present. It is imperative to differentiate between physical button presses from a patient in pain from those requests that the PCAg device actually generates. The PCAg device should only fulfill requests within a normal threshold. If a PCAg device fulfills requests too frequently, then malicious behavior is present. In addition, another scenario when the PCAg device acts as a trustee and the VSM acts as a monitor occurs as the PCAg device administers a pain relief dosage with a certain infusion rate to a cardiac patient who has an implanted CD in the defibrillation mode. This state includes two components: the infusion rate and the CD mode. As the device being evaluated transitions from one state to another, the VSM can determine if both states are either safe or malicious, depending on the infusion rate and CD device mode. To illustrate further, when a PCAg device has an infusion rate R in the range (0,100%) and the cardiac device mode, M, is in defibrillation, the monitor can check to see if $(R0, M0)$ and $(R1, M1)$ are both safe states.

4.4 Recognizing State Components and Ranges

In order to complete the synthesis process, our research quantized the continuous components related to the sensors and actuator readings based on an integer scale that fell within admissible ranges. The complete list of the admissible ranges for the state components is essential.

Table 4.5: State component in the designed state machine

Name	Attribute	Range	Component	Device
Analgesic Request	Reading	True, False	Sensor	PCAg
Pulse	Reading	[0,240 bpm]	Sensor	VSM
Respiration	Reading	[0,60 bpm]	Sensor	VSM
Analgesic Request Rate	Reading	[0,4/hour]	Sensor	PCAg
Blood Pressure	Reading	[0,240 mmHg] × [0,160 mmHg]	Sensor	VSM
Oxygen saturation	Reading	[0,100%]	Sensor	VSM
Temperature	Reading	[32,42 °C]	Sensor	VSM
Analgesic Infusion Rate	Control	[0,100%]	Actuator	PCAg
Mode	Control	Passive, Pacemaker, Defibrillator	Sensor	CD
Pacemaker Frequency	Control	[0,240 bpm]	Actuator	CD
Insulin Request	Reading	True, False	Sensor	CGM
Insulin Request Rate	Reading	[0,4/hour]	Sensor	CGM
Glucose	Reading	[0,200 mg/dL]	Sensor	CGM
Insulin Infusion Rate	Control	[0,100%]	Actuator	CGM

4.4.1 VSM Device

In VSM devices, the state components include pulse rate, respiration rate, temperature, blood pressure, and pacemaker frequency. According to the admissible ranges for these states, the normal pulse rate range falls within [0, 240 bpm] yielding 241 values; the normal respiration rate range falls within [0, 60 bpm] yielding 61 values; the normal temperature range falls within [32, 42 °C] yielding 11 values; the normal blood pressure range falls within [0, 240 mmHg] × [0, 160 mmHg] yielding 241 × 161 values; and the normal pacemaker frequency range falls within [0, 240 bpm] yielding 241 values. Therefore, the total values of all the states is equivalent to: $241 \times 161 \times 241 \times 161 \times 101 \times 101 \times 241 \times 241 \times 61 \times 61 \times 11 \times 11 = 4.16 \times 1023$ states.

As demonstrated, the resulting states' space for each device is huge, and a reduction technique is necessary to reduce the total number of states. From a medical point of view; our research reduces the states' space by eliminating the values of specific state components that are related to each device. For example, in a VSM device, we define three values that are relevant for pulse rate, respiration rate, temperature,

blood pressure, and pacemaker frequency. These are normal, beyond warning threshold, and beyond unsafe threshold. In addition to the standard states' components connected to each device, we monitor specific state components for each device.

4.4.2 PCAg Device

Similarly, in PCAg devices, the state components encompass of analgesic request, respiration rate, analgesic request rate, and analgesic infusion rate. According to the normal admissible ranges for these states: an analgesic request is [0, 1] yielding 2 values; the respiration rate is [0, 240 bpm] yielding 241 values; the analgesic request rate is in [0, 4] yielding 5 values; and the analgesic infusion rate is [0, 100] yielding 101 values. Therefore, the total value of all the states is equivalent to : $2 \times 241 \times 5 \times 61 \times 5 \times 101 \times 3 = 4.454 \times 10^7$ states.

As demonstrated, the resulting states' space for each device is huge, and a reduction technique is necessary to reduce the total number of states. From a medical point of view; our research reduces the states' space by eliminating the values of specific state components that are related to each device. For example, in the PCAg device, we define the values that are relevant for an analgesic infusion rate. Also, the two values that are relevant for the analgesic request are zero or greater than zero, and the two values that are relevant for insulin request are zero or greater than zero.

4.4.3 CD Device

Likewise, in CD devices, the state components comprise of the pacemaker frequency, oxygen saturation level, pulse rate, and CD device mode. According to the normal admissible ranges for these states, the pacemaker frequency range is [0, 240 bpm] yielding 241 values; the oxygen saturation level range is [0,100] yielding 241 values; the pulse rate's range is [0, 240 bpm] yielding 241 values; and in an active CD

device the mode is either defibrillation or pacemaker which yields 3 values (taking in consideration the active mode of the CD device). Therefore, the total value of all the states is equivalent to: $241 \times 101 \times 241 \times 3\times = 1.760 \times 10^7$ states.

The resulting states' space for each device is huge, and a reduction technique is necessary to reduce the total number of states. Our research reduces the states' space by eliminating the values of specific state components that are related to each device. In a CD device, we define the values that are relevant for oxygen saturation levels. Also, we define the values that are relevant for the CD device's mode.

4.4.4 CGM Device

In the same manner, for CGM devices, the state components consist of an insulin request, glucose level, insulin request rate, insulin infusion rate, and pulse rate. The admissible ranges for these states incorporate an insulin request that is either true or false and falls within the range of [0, 1] yielding 2 values; the glucose level is in the range of [0, 200] yielding 201 values; insulin request rate is in the range of [0, 2] yielding 3 values; insulin infusion rate is in the range of [0,100] yielding 101 values; and the pulse is in the range of [0, 240 bpm] yielding 241 values. Therefore, the total value of all the states is equivalent to: $3 \times 201 \times 3 \times 101 \times 241 = 9.785 \times 10^6$ states.

As can be seen, the resulting states' space for the CGM device is also huge, and a reduction technique is necessary to reduce the total number of states.

Through our research, we found that the resulting states' space for the VSM device is equivalent to $3 \times 3 \times 3 \times 3 \times 3 = 243$ states. In the PCAg device, it is equivalent to $2 \times 3 \times 3 \times 3 \times 2 \times 3 = 324$ states, in the CD device it is equivalent to $3 \times 3 \times 3 \times 3 = 81$ states. Lastly, the resulting states' space in the CGM device is equivalent to $2 \times 3 \times 3 \times 2 \times 3 = 108$ states.

Table 4.6: Number of states and cause for each particular state

State	VSM	PAC	CD	CGM	Cause
Safe	1	50	4	3	The monitor and trustee readings match for all the specified related components as described in Table 4.5
Warning	31	80	23	57	The monitor and trustee readings differ by more than the warning margin for at least one of the specified related components but not more than the unsafe threshold for any of the components as described in Table 4.5
Unsafe	211	194	54	48	At least one of the specified related components differs by more than the unsafe threshold as described in Table 4.5
Idle	1	1	1	1	Initial State of the system
Using ranges	244	325	82	109	The ranges were limited according to the medical prospect
Without ranges	4.016×10^{23}	4.454×10^{7}	1.760×10^{7}	9.785×10^{6}	The ranges were comprised of wide ranges of Natural numbers.

CHAPTER 5: RESULTS, DISCUSSIONS AND COMPARISONS

In this chapter, we present the principal findings of the designed experiments, both the machine learning based IDS designs in software and the software and hardware-based MCPS IDS co-designs.

5.1 Machine Learning Based IDS Results

All the work for the experiments was carried out using Intel Core i7 3.30 GHz, 16 GB RAM system running Windows 10 and the Waikato Environment for Knowledge Analysis (Weka) software.

The proposed feature selection ideas were based on four groups of selected features tested on the AWID dataset. The results obtained from the analysis of the performance of classifiers with 32-FSG, 10-FSG, 7-FSG and 5-FSG are summarized hereafter. To evaluate the performance of these selected feature groups, several machine learning classification algorithms were applied to train and test the data. The classifiers were trained and tested and the average accuracy was recorded. We tested different classifiers performance, since a certain classifier might yield better performance compared to others. There is no best machine learning classifier because a set of machine learning models that work very well in one domain may work poorly in another. Hence, to ensure that the IDS model is pragmatic and feasible in real-world,

we performed tests using various types of classifiers and determined the best one that covers the wide variety of attacks included in the AWID. Moreover, based on our survey, no previous studies have considered cross validation approaches for testing the different classifiers. The chosen classifiers applied to the proposed selected feature set groups in this study with AWID include OneR, ZeroR, Random Forest, Random Tree, Bagging, Logit Boost, Simple Logistic J48 , and NN, which are among the most commonly used ones based on our survey.

The feature reduction approaches were examined on the CICIDS2017 dataset as well as AWID, and balancing techniques were applied along with various classifiers on different benchmark IDS datasets (e.g. AWID, CICIDS2017 and CIDDS-001).

5.1.1 Performance Evaluation Metrics

In this work, we evaluate performance with multi-class and binary class datasets. In the literature, most of the performance measures are designed only for two-class problems. Based on the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) values, the following set of metrics are used to evaluate the performance of IDS classifiers.

This study used various performance metrics to evaluate the performance of the proposed ideas for IDS, including False Alarm Rate (FAR), Matthews Correlation Coefficient (MCC) [12], F-Measure [19], Detection Rate (DR), and Accuracy (Acc) as well as the processing time (in seconds). The definition of these metrics are provided below.

(1) False Alarm Rate (FAR) is a common term which encompasses the number of normal instances incorrectly classified by the classifier as an attack and can be estimated through Equation 5.1.

$$FAR = \frac{FP}{TN + FP} \tag{5.1}$$

(2) Accuracy (Acc) is defined as the ability measure of the classifier to correctly classify an object as either normal or attack. The Accuracy can be defined using Equation 5.2.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

(3) Detection Rate (DR) indicates the number of attacks detected divided by the total number of attack instances in the dataset. DR can be estimated by Equation 5.3.

$$DR = \frac{TP}{TP + FN} \tag{5.3}$$

(4) F-measure (F-M) is a score of a classifier's accuracy and is defined as the weighted harmonic mean of the precision and recall measures of the classifier. F-Measure is calculated using Equation 5.4.

$$F\text{-}Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

(5) MCC is the Matthews Correlation Coefficient and can be thought as a measure of the quality of classification [12]. MCC is calculated using Equation 5.5.

$$MCC = \frac{(TP \times TN)(FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.5}$$

(6) Moreover, this dissertation studies the Kappa statistic metric [16] for multi-class classification. According to Mikel et al. [15], one way to calculate the Kappa statistic metric is by making use of the resulting confusion matrix in the classification task. From this matrix, Kappa statistic is computed from Equation 5.6 as follows:

$$\text{Kappa} = \frac{n \sum_{i=1}^{m} hi_i - \sum_{i=1}^{m} Tr_i Tc_i}{n^2 - \sum_{i=1}^{m} Tr_i Tc_i} \tag{5.6}$$

where hi_i is the cell count in the main diagonal (the number of true positives for each class), n is the number of samples, m is the number of class labels and Tr_i and Tc_i are the total number of rows and columns, respectively.

(7) Precision (Pre.) represents the number of positive predictions divided by the total number of positive class values predicted. It is considered as a measure for the classifier exactness. A low value indicates large number of False Positives. The precision is calculated using Equation 5.7.

$$Precision = \frac{TP}{TP + FP} \tag{5.7}$$

(8) Recall (Rec.) is the number of True Positives divided by the number of True Positives and the number of False Negatives. Recall is considered as a measure of a classifier completeness such that a low value of recall realizes many False Negatives [214, 215]. Recall is estimated through Equation 5.8.

$$Recall = \frac{TP}{TP + FN} \tag{5.8}$$

(9) The geometric [216] (G-mean) for multi-class problems is a higher root of the product of recall for each class [217], which is estimated using Equation 5.9:

$$\left(\prod_{i=1}^{K} R_i\right)^{\frac{1}{K}} \tag{5.9}$$

where R_i denotes the recall of class C_i.

5.2 Multi-class Combined Performance Metric

Basically, the overall accuracy is used to measure the effectiveness of a classifier. Unfortunately, in presence of imbalanced data, this metric may fail to provide adequate information about the performance of the classifier. Furthermore, the method is very sensitive to the class distribution and might be misleading in some way. Hamed et al. [20] proposed a combined performance metric to compare multiple binary classifier systems. However, their solution neglects class distribution and can work only for binary classification systems.

Table 5.1: Proposed $Combined_{Mc}$ metric calculation pseudo-code

Calculate $Combined_{Mc}$ with respect to Class Distribution
Feed Confusion Matrix CM
For i =1 to C
Calculate the total number of **FP** for C_i as the sum of values in the i^{th} column excluding **TP**
Calculate the total number of **FN** for C_i as the sum of values in i^{th} row excluding **TP**
Calculate the total number of **TN** for C_i as the sum of all columns and rows excluding i^{th} row and column
Calculate the total number of **TP** for C_i as the diagonal of the i^{th} cell of CM
Calculate the total number of instances for C_i as the sum of the i^{th} row
Calculate the total number of instances for C_i as the sum of the i^{th} row
Calculate the total number of instances in the dataset as the sum of all rows
Calculate Acc using Eq. 5.2, DR using Eq. 5.3, and FAR using Eq. 5.1 for each class C_i
Calculate the distribution of each C_i using Eq. 5.11
i ++
Calculate $Combined_{Mc}$ using Eq. 5.10

In this dissertation, we propose the Multi-Class Combined performance metric $Combined_{Mc}$ with respect to class distribution so that it can compare multiple multi-class classification systems as well as binary class systems through incorporating four metrics together (Accuracy 5.2, Detection Rate 5.3, FAR 5.1, and class distribution 5.11). The Multi-Class Combined performance metric can be estimated using the following equation 5.10.

$$Combined_{Mc} = \sum_{i=1}^{C} \lambda_i \left(\frac{Acc_i + DR_i}{2} - FAR_i \right) \tag{5.10}$$

where C is number of classes, and λ_i is the class distribution, which can be estimated using the following formula:

$$Class\ Distribution = \frac{Total\ number\ of\ instances\ in\ each\ class}{Total\ number\ of\ instances\ in\ the\ dataset} \tag{5.11}$$

The result of this metric will be a real value between -1 and 1; in other words, $Combined_{Mc} \in [1, -1]$; where -1 corresponds to the worst overall system performance and 1 corresponds to the best overall system performance. Table 5.1 illustrates the pseudo-code for calculating this proposed combined metric.

5.3 Results and Discussion - AWID

5.3.1 32-FSG

As seen from Table 5.2, the highest result of accuracy with 32-FSG was 99.64% for the Random Forest algorithm, whereas, the lowest was 66.64% for Simple Logistic. Here, the classifier fails to classify some of the attacks included in AWID. This algorithm is a function-based algorithm and changing the activation function may enhance the Simple Logistics' accuracy. Table 5.2 also compares the FP rate results from the analysis of the classifiers. The highest FP rate was 0.933 for the Naïve Bayes, whereas, the lowest FP rate was 0.04 for the Logit Boost classifier. The classifiers also resulted in significantly high Precision and Recall values. Logit boost reported the highest Precision and Recall value of 0.996. It is apparent from Table 5.2 that the highest result of the F-Measure value was 0.996 for the Logit Boost classifier. This is compatible with the resulted Accuracy, Precision and Recall for the same classifier. Moreover, the area under ROC curve is another good metric to compare the performance of the classifiers. Here, higher values indicate better performance. The obtained results indicate that the ROC curve area for ZeroR is the lowest value of 0.5. Nonetheless, Random Forest and Logit Boost show high ROC curve areas of 0.993 and 0.991, respectively. Moreover, the Area under Precision-Recall curves are better to highlight the differences between models for highly imbalanced datasets.

5.3.2 10-FSG

This subsection presents the results for the second feature set group. First, a comparison of the resulted accuracy is shown in Table 5.3. For the 10-FSG the Naïve Bayes achieved the highest accuracy of 99.14%. Multi- layer Perceptron CS, which is a function based algorithm, achieved an accuracy of 91.21% with the 10-FSG. Second,

Table 5.2: Classifiers evaluation of AWID-ATK-R with 32-FSG

Classifier	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Time
J48	99.42	0.994	0.1	0.994	0.994	0.994	0.926	0.959	0.991	469.91
OneR	96.17	0.962	0.909	0.93	0.962	0.946	0.149	0.512	0.891	11.13
Random Forest	99.64	0.966	0.077	0.995	0.966	0.995	0.956	0.993	0.998	2277.51
Naïve Bayes	96.20	0.962	0.933	0.954	0.962	0.946	0.093	0.532	0.92	9.8
Bagging	99.15	0.992	0.111	0.991	0.992	0.991	0.889	0.943	0.988	1326.53
Multi-Layer Perceptron CS	98.05	0.981	0.343	0.976	0.981	0.977	0.788	0.895	0.98	17256
Simple Logistic	66.64	0.666	0.639	0.987	0.666	0.789	0.26	0.612	0.95	16830.26

Table 5.3: Classifiers evaluation of AWID-ATK-R with 10-FSG

Classifier	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Time
J48	96.33	0.963	0.94	0.929	0.963	0.946	0.109	0.603	0.927	53.45
OneR	96.17	0.962	0.909	0.93	0.962	0.946	0.149	0.512	0.891	3.2
Random Forest	96.40	0.964	0.939	0.93	0.964	0.946	0.151	0.826	0.981	1197.08
Naïve Bayes	99.14	0.991	0.083	0.989	0.991	0.99	0.882	0.927	0.989	3.24
Bagging	96.40	0.964	0.906	0.932	0.964	0.947	0.214	0.772	0.967	298.53
Multi-Layer Perceptron CS	91.21	0.912	0.514	0.944	0.912	0.928	0.296	0.886	0.956	4564.3
Simple Logistic	95.51	0.955	0.858	0.933	0.955	0.943	0.159	0.966	0.988	2049.28

Table 5.4: Classifiers evaluation of AWID-ATK-R with 7-FSG

Classifier	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Time
J48	98.82	0.988	0.109	0.984	0.988	0.986	0.913	0.908	0.974	37.61
OneR	96.17	0.962	0.909	0.93	0.962	0.946	0.149	0.512	0.891	2.52
Random Forest	99.31	0.993	0.107	0.993	0.993	0.992	0.921	0.971	0.991	582.63
Bagging	99.44	0.994	0.099	0.994	0.994	0.924	0.924	0.877	0.972	321.79
Multi-Layer Perceptron CS	96.09	0.963	0.858	0.932	0.963	0.947	0.272	0.88	0.948	3518.46
Simple Logistic	94.93	0.949	0.86	0.932	0.949	0.94	0.122	0.959	0.9	1501.04
Naïve Bayes	99.3	0.993	0.083	0.991	0.993	0.992	0.902	0.967	0.99	2.18

Table 5.5: Classification evaluation of AWID-ATK-R with 5-FSG

Classifier	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Time
Bagging	99.44	0.994	0.099	0.994	0.994	0.994	0.924	0.877	0.972	260.2
Random Forest	99.35	0.993	0.106	0.993	0.993	0.993	0.922	0.971	0.991	538.82
Naïve Bayes	99.30	0.993	0.083	0.991	0.993	0.992	0.902	0.965	0.99	1.52
J48	98.82	0.988	0.109	0.948	0.988	0.986	0.913	0.908	0.974	23.67
OneR	96.17	0.962	0.909	0.93	0.962	0.946	0.149	0.512	0.891	1.71
Multi-Layer Perceptron CS	96.09	0.961	0.727	0.936	0.961	0.948	0.415	0.934	0.957	2133.21
Simple Logistic	94.93	0.949	0.86	0.932	0.949	0.94	0.122	0.959	0.9	1516.93

as presented in Table 5.3, the FP Rates of different classifiers are compared. What stands out from this Table is that, the majority of the classifiers achieved a high FP Rate, which is not a good indication. However, the Naïve Bayes, Logit Boost and AdaBoost achieved lower values of 0.083, 0.314, and 0.132, respectively for the FP Rate. In addition, the F-Measure is an important metric that has the ability to reflect whether or not the values of precision or recall excel from each other. For the 10-FSG, all the classifier models reported an F-measure value greater than 0.928.

5.3.3 7-FSG

In this section, we present the principal findings of the experiments relevant to 7-FSG. Table 5.4 presents a comparison of the Accuracies, FP rate, Precision, Recall, ROC Area, PRC Area, F-Measure, MCC, and Time to build the model among the classifier models. The highest values are shown in purple.

5.3.4 5-FSG

In this section, we present the principal findings of the experiments relevant to 5-FSG. Table 5.5 presents a comparison of the Accuracies, FP rate, Precision, Recall, ROC Area, PRC Area, F-Measure, MCC, and Time to build the model among the classifier models. The highest values are shown in purple.

5.3.5 7-FSG Cross Validation

To further examine the 7-FSG we ran the experiments using the 10-fold cross validation approach to validate the obtained results. The cross-validation results are set out in Table 5.6. The highest values are highlighted. The Random Forest reports an accuracy of 99.99%.

Table 5.6: Performance evaluation of the cross validation approach using 7-FSG

Classifier	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Time
Random Forest	99.99	1	0	1	1	1	1	1	1	178.53
J48	99.99	1	0	1	1	1	1	1	1	6.58
Simple Logistic	98.92	0.989	0.049	0.989	0.989	0.988	0.936	0.996	0.992	407.77
Bagging	99.99	1	0	1	1	1	1	1	1	41.81
Naïve Bayes	97.37	0.974	0.11	0.986	0.974	0.978	0.853	0.995	0.998	0.72
OneR	96.49	0.965	0.345	0.939	0.956	0.951	0.739	0.812	0.937	0.83
Multi-Layer Perceptron CS	99.74	0.997	0.019	0.997	0.997	0.997	0.984	0.996	0.998	1202.98

Table 5.7: Performance evaluation of the cross validation approach using 5-FSG

Classifier	Accuracy (%)	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Time
J48	99.99	1	0	1	1	1	0.999	1	1	5.51
OneR	96.50	0.965	0.342	0.939	0.965	0.951	0.739	0.812	0.937	0.89
Random Forest	99.99	1	0	1	1	1	1	1	1	1
Naïve Bayes	97.33	0.974	0.011	0.986	0.974	0.978	0.853	0.995	0.998	0.59
Bagging	99.99	1	0	1	1	1	1	1	1	298.41
Simple Logistic	98.76	0.988	0.08	0.983	0.988	0.985	0.917	0.995	0.991	1407.3
Multi-Layer Perceptron CS	99.65	0.997	0.021	0.997	0.997	0.996	0.981	0.996	0.997	1603.38

5.3.6 5-FSG Cross Validation

To further examine the 5-FSG, we ran the experiments using the 10-fold cross validation approach to validate the obtained results. The cross-validation results are set out in Table 5.7. Theis Tables illustrates that the maximum obtained accuracy was 99.99% using the Random Forest classifier. Furthermore, the resulted FP Rate for J48 and Bagging was the lowest value of 0, which is the best result. Conversely, the OneR classifier achieved the highest FP Rate of 0.342. The Precision values range from a maximum value of 1 to a minimum value of 0.939.

Furthermore, a visualization of the AWID-ATK-R-Trn and AWID-ATK-R-Tst based on 2 PCA Components analysis is shown in Figures 5.1 and Figure 5.2, respectively.

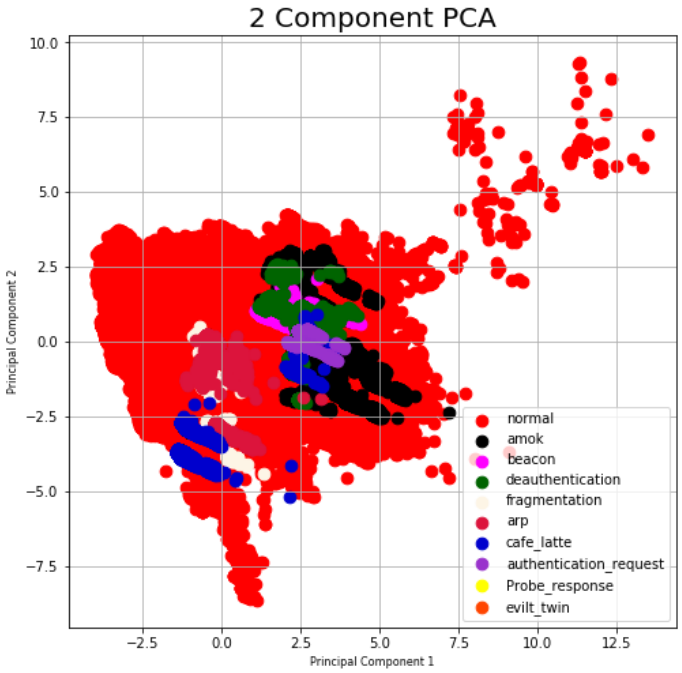

Figure 5.1: 2D Visualization of PCA on AWID-ATK-R-Trn

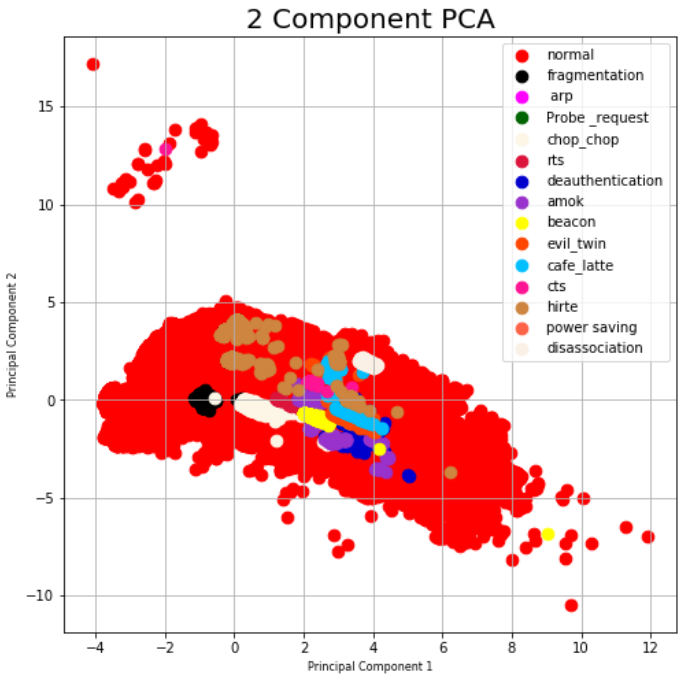

Figure 5.2: 2D Visualization of PCA on AWID-ATK-R-Tst

Table 5.8: Performance evaluation of 32-FSG under imbalanced class distribution handling approaches

	Original Distribution			RWR			RWoR			Class Balancer			SMOTE		
Classifiers	Accuracy	ROC Area	PRC Area	Accuracy	Roc Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area
32-FSG 8 Classes															
Random Forest	0.996	1.000	0.999	0.995	1.000	0.999	0.995	1.000	0.996	0.996	1.000	0.999	0.776	0.969	0.994
Naïve Bayes One VS All	0.961	0.672	0.968	0.963	0.980	0.944	0.963	0.618	0.966	0.964	0.561	0.956	0.963	0.923	0.989
MultiBoost with J48	0.995	0.999	0.999	0.994	1.000	0.998	0.997	1.000	0.998	0.997	1.000	0.998	0.989	1.000	0.999
32-FSG 10 Classes															
Random Forest	0.996	1.000	0.999	0.995	1.000	0.999	0.987	0.998	0.640	0.996	1.000	0.999	0.964	0.999	0.997
Naïve Bayes One VS All	0.963	0.992	0.972	0.973	0.838	0.960	0.961	0.672	0.968	0.962	0.765	0.0972	0.947	0.977	0.989
MultiBoost with J48	0.988	0.999	0.999	0.996	1.000	0.999	0.9872	1.000	0.9980	0.998	1.000	0.999	0.964	0.999	0.997

Table 5.9: Performance evaluation of 10-FSG under imbalanced class distribution handling approaches

	Original Distribution			RWR			RWoR			Class Balancer			SMOTE		
Classifiers	Accuracy	ROC Area	PRC Area	Accuracy	Roc Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area
10-FSG 8 Classes															
Random Forest	0.982	0.993	0.999	0.993	1.000	0.914	0.987	0.999	0.999	0.964	0.998	0.998	0.953	0.986	0.994
Naïve Bayes One VS All	0.958	0.615	0.989	0.992	0.597	0.961	0.955	0.965	0.988	0.964	0.645	0.979	0.957	0.964	0.995
MultiBoost with J48	0.958	0.615	0.995	0.992	0.914	0.914	0.955	0.965	0.988	0.982	0.993	0.999	0.951	0.969	0.993
10-FSG 10 Classes															
Random Forest	0.952	0.963	0.977	0.996	1.000	0.999	0.990	0.990	0.998	0.989	0.998	0.998	0.786	0.812	0.979
Naïve Bayes One VS All	0.989	1.000	0.998	0.960	0.727	0.965	0.910	0.994	0.971	0.989	0.956	0.998	0.618	0.575	0.932
MultiBoost with J48	0.988	0.885	0.980	0.992	0.987	0.997	0.990	0.971	0.994	0.988	0.956	0.998	0.618	0.575	0.932

Table 5.10: Performance evaluation of 7-FSG under imbalanced class distribution handling approaches

	Original Distribution			RWR			RWoR			Class Balancer			SMOTE		
Classifiers	Accuracy	ROC Area	PRC Area	Accuracy	Roc Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area
7-FSG 8 Classes															
Random Forest	0.963	0.735	0.957	0.964	0.823	0.967	0.966	0.882	0.971	0.963	0.735	0.957	0.961	0.876	0.966
Naïve Bayes One VS All	0.948	0.928	0.994	0.963	0.924	0.992	0.933	0.927	0.994	0.894	0.924	0.992	0.742	0.972	0.979
MultiBoost with J48	0.946	0.937	0.978	0.962	0.523	0.931	0.963	0.547	0.936	0.963	0.520	0.931	0.961	0.588	0.938
7-FSG 10 Classes															
Random Forest	0.962	0.789	0.966	0.963	0.901	0.982	0.960	0.940	0.984	0.962	0.897	0.977	0.962	0.767	0.964
Naïve Bayes One VS All	0.842	0.929	0.973	0.931	0.764	0.965	0.937	0.693	0.952	0.905	0.5189	0.925	0.843	0.943	0.972
MultiBoost with J48	0.961	0.560	0.931	0.966	0.554	0.925	0.842	0.885	0.970	0.960	0.560	0.931	0.961	0.684	0.940

Table 5.11: Performance evaluation of 5-FSG under imbalanced class distribution handling approaches

	Original Distribution			RWR			RWoR			Class Balancer			SMOTE		
Classifiers	Accuracy	ROC Area	PRC Area	Accuracy	Roc Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area	Accuracy	ROC Area	PRC Area
5-FSG 8 Classes															
Random Forest	0.994	0.984	0.995	0.994	0.996	0.996	0.994	0.984	0.995	0.994	0.984	0.995	0.993	0.967	0.991
Naïve Bayes One VS All	0.963	0.615	0.962	0.717	0.964	0.985	0.993	0.958	0.988	0.994	0.971	0.994	0.803	0.975	0.871
MultiBoost with J48	0.988	0.952	0.992	0.994	0.972	0.994	0.995	0.972	0.994	0.935	0.573	0.964	0.992	0.933	0.985
5-FSG 10 Classes															
Random Forest	0.989	0.995	0.964	0.960	0.964	0.994	0.951	0.976	0.994	0.987	0.978	0.994	0.989	0.997	0.990
Naïve Bayes One VS All	0.960	0.994	0.992	0.635	0.991	0.991	0.891	0.977	0.992	0.989	0.956	0.998	0.861	0.976	0.877
MultiBoost with J48	0.995	0.992	0.998	0.882	0.987	0.986	0.923	0.963	0.993	0.995	0.992	0.998	0.989	0.999	0.989

5.3.7 Handling Imbalanced Class Distributions in AWID

This section presents the effect of applying imbalanced class handling approaches on AWID. The effect of applying different balancing techniques on the 32-FSG, 10-FSG, 7-FSG, and 5-FSG is displayed in Tables 5.8, 5.9, 5.10, and 5.11, respectively.

5.3.8 Performance Comparison - AWID

The following paragraph compares our experiments with previous related works that used the AWID dataset. We must note that comparison with other studies is challenging because of the differences in the Full and Reduced version of the AWID dataset.

Thanthrige et al. [95] [89] achieved the highest accuracy of 94.97% using Random Forest with 41 features (attributes). In addition, when they reduced the attributes to 10, the results showed that the accuracy decreased from 94.97% to 92.29% for Random Forest. This is while our study achieved 99.99% accuracy using Random Forest with 7-FSG and 5-FSG. We were also able to achieve a minimum accuracy of 94.93% with the Simple Logistic classifier using 7-FSG and a higher accuracy of 99.44% with the Bagging approach using 7-FSG.

Tables 5.12 highlights a comparison of previous studies with our study on the AWID dataset. The table shows that our study achieved the maximum accuracy of 99.99%. As demonstrated from the results, the selected attributes (features) impact positively upon our classifier models. To be specific, attributes such as epoch.time, frame.time.relative, and wlan.fc.pwrmgt are essential and effective to detect attacks such as Caffe latte, Hirte, Honeypot and Evil twin [88] [95]. This is while large features sets may result in over-fitting problems, and optimal attribute/feature set selection reduces the processing time, which is crucial for real-time applications.

Table 5.12: Summary of the work related to AWID dataset

Study	Dataset	Approach	Attributes	Classes	Accuracy (%)
Alotaibi [218]	AWID-CLS-R	Majority voting	21	4	92.36
Kolias [2]	AWID-CLS-R	J48	20	4	96.2
Aminanto [98]	AWID-CLS-R	ANN	154	2	99.86
Aminanto [97]	AWID-CLS-R	SAE	154	4	97.7
Kaleem [105]	AWID-CLS-R	ANN	7	2	99.3
Thanthrige [89]	AWID-CLS-R	Random Forest	111	4	94.83
Thanthrige [89]	AWID-CLS-R	Random Tree	41	4	95.12
Thanthrige [89]	AWID-CLS-R	J48	10	4	92.44
Thing [106]	AWID-CLS-R	Deep learning	154	4	98.67
Thanthrige [95]	AWID-ATK-R	Random Tree	111	4	94.58
Thanthrige [95]	AWID-ATK-R	Random Forest	41	4	94.97
Thanthrige [95]	AWID-ATK-R	Random Forest	10	4	92.29
Our Study	AWID-ATK-R	Random Forest	32	15	99.64
	AWID-ATK-R	Naïve Bayes	10	15	99.14
	AWID-ATK-R	Logit Boost	7-a	15	99.56
	AWID-ATK-R	Logit Boost	5-a	15	99.53
	AWID-ATK-R	Random Forest, J48, Bagging	7-Cross Validation	15	99.99
	AWID-ATK-R	Random Forest, J48, Bagging	5-Cross Validation	15	99.99

5.4 Results and Discussion - CICIDS2017

Extensive simulations have been performed on the CICIDS2017 dataset. All the simulations were carried out using an Intel-Core i7 with 3.30 GHz and 32 GB RAM, running Windows 10. The results highlight the advantages of feature dimensionality reduction and balancing on CICIDS2017.

From the research efforts in this work, we were able to reduce the dimensionality of the features in CICIDS2017 from 81 features to 10 features while maintaining a high accuracy in multi-class and binary class classification using the Random Forest classifier. The findings are discussed in following subsections.

5.4.1 Binary class Classification

The study evaluates the performance of binary classification (Bc) in terms of Acc, FAR, DR and F-M. Tables 5.13 and 5.14 display the summary of the results obtained. Table 5.13 highlights the results of the dimensionality reduction of the features in CICIDS2017 from 81 features to 10 features obtained using PCA, whereas

Table 5.13: Performance evaluation of binary classification using PCA

No.of Features	$(PCA-RF)_{Bc-X}$					$(PCA-BN)_{Bc-X}$					$(PCA-LDA)_{Bc-X}$					$(PCA-QDA)_{Bc-X}$				
	Acc	FAR	DR	F-M	CM_{Bc}	Acc	FAR	DR	F-M	CM_{Bc}	Acc	FAR	DR	F-M	CM_{Bc}	Acc	FAR	DR	F-M	CM_{Bc}
81	0.995	0.002	0.984	0.996	0.987	0.975	0.025	0.976	0.976	0.951	0.937	0.254	0.811	0.937	0.846	0.782	0.237	0.978	0.807	0.632
70	0.997	0.001	0.989	0.997	0.991	0.970	0.029	0.966	0.970	0.938	0.947	0.274	0.821	0.947	0.856	0.792	0.247	0.988	0.817	0.642
64	0.996	0.002	0.986	0.996	0.988	0.969	0.029	0.966	0.970	0.968	0.947	0.027	0.820	0.947	0.466	0.793	0.245	0.988	0.818	0.891
59	0.997	0.0017	0.989	0.997	0.991	0.968	0.030	0.963	0.969	0.934	0.947	0.027	0.823	0.947	0.858	0.794	0.244	0.988	0.819	0.646
50	0.996	0.0017	0.989	0.997	0.991	0.971	0.025	0.959	0.972	0.939	0.945	0.028	0.814	0.945	0.880	0.809	0.226	0.988	0.832	0.838
40	0.997	0.001	0.990	0.997	0.991	0.974	0.021	0.953	0.974	0.941	0.944	0.032	0.829	0.945	0.856	0.808	0.226	0.981	0.831	0.646
30	0.997	0.001	0.989	0.997	0.990	0.979	0.026	0.956	0.971	0.703	0.944	0.030	0.821	0.945	0.852	0.830	0.198	0.974	0.849	0.703
20	0.996	0.001	0.989	0.997	0.991	0.965	0.031	0.948	0.966	0.926	0.878	0.025	0.396	0.862	0.612	0.717	0.332	0.969	0.754	0.511
10	0.996	0.001	0.988	0.997	0.991	0.952	0.036	0.897	0.953	0.889	0.869	0.028	0.363	0.852	0.588	0.712	0.048	0.966	0.749	0.911

Table 5.14 displays the results of the dimensionality reduction of the features in CICIDS2017 from 81 features to 59 features utilizing AE. In the tables, X denotes the number of features after reduction.

The DR metric revealed that $(PCA-RF)_{Bc-10}$ is able to detect 98.8% of the attacks. In the same manner, $(PCA-RF)_{Bc-10}$ achieved an F-Measure of 0.997. Moreover, $(AE-RF)_{Bc-59}$ is able to detect 98.5% of the attacks.

Figure 5.3 highlights the achieved detection rates resulted from the dimensionality reduction using PCA, whereas, Figure 5.4 shows the achieved detection rate using the reduced features set by AE. From Figures 5.3 and 5.4, it is apparent that Random forest, QDA and Bayesian Network reported significantly higher detection rates than the LDA for the reduced feature dimensionality of CICIDS2017 using the PCA approach. The results from the classification using different classifiers assures that our reconstructing of new feature representation was good enough to achieve an overall accuracy of 98.5% with 59 features in binary classification using Random Forest from AE.

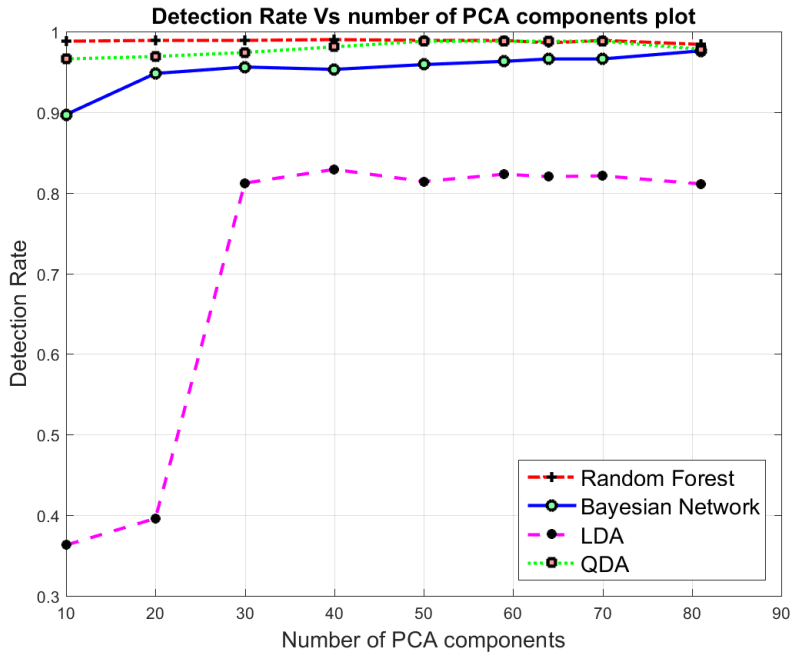

Figure 5.3: Binary class classification: detection rate in terms of number of components using PCA

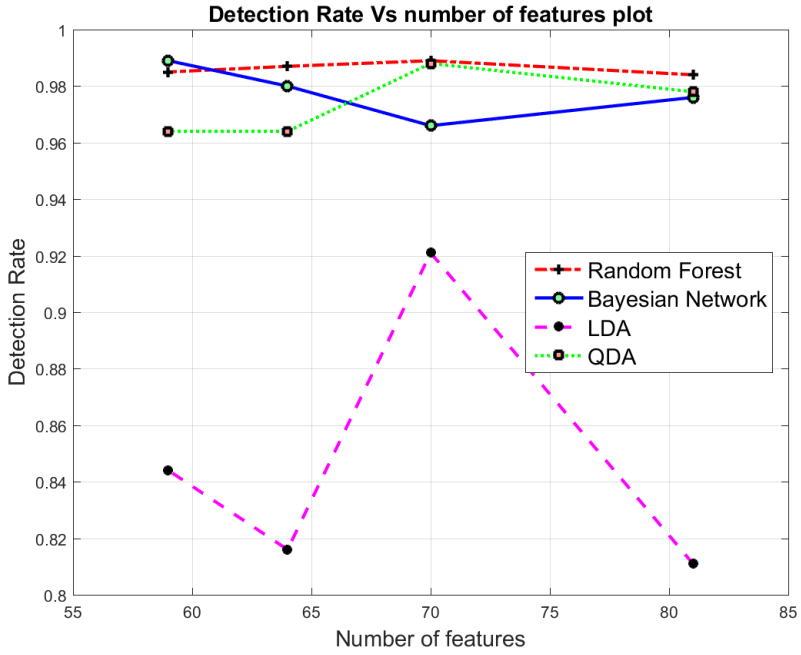

Figure 5.4: Binary class classification: detection rate in terms of number of features using AE

Table 5.14: Performance evaluation of binary classification using AE

	$(AE-RF)_{Bc-X}$					$(AE-BN)_{Bc-X}$					$(AE-LDA)_{Bc-X}$					$(AE-QDA)_{Bc-X}$				
	Acc	FAR	DR	F-M	CM_{Bc}	Acc	FAR	DR	F-M	CM_{Bc}	Acc	FAR	DR	F-M	CM_{Bc}	Acc	FAR	DR	F-M	CM_{Bc}
81	0.995	0.002	0.984	0.996	0.987	0.975	0.025	0.976	0.976	0.951	0.937	0.254	0.811	0.937	0.846	0.782	0.237	0.978	0.807	0.632
70	0.997	0.001	0.989	0.997	0.991	0.970	0.029	0.966	0.970	0.938	0.947	0.274	0.921	0.947	0.856	0.792	0.247	0.988	0.817	0.642
64	0.996	0.002	0.987	0.996	0.973	0.974	0.026	0.980	0.975	0.948	0.947	0.027	0.816	0.946	0.854	0.935	0.070	0.964	0.939	0.894
59	0.995	0.002	0.985	0.996	0.988	0.975	0.027	0.989	0.976	0.955	0.948	0.030	0.844	0.949	0.866	0.942	0.063	0.964	0.944	0.890

5.4.2 Multi-class Classification

The study used the Acc, F-M, FPR, TPR, Precision, Recall, and the Combined multi-class metrics to evaluate the performance of multi-class classification. Tables 5.15 and 5.16 display the summary of the results obtained for the dimensionality reduction of the features for CICIDS2017 from 81 to 10 using PCA, and from 81 to 59 using AE, respectively. In the tables, X denotes the number of features after reduction.

Figure 5.5 presents the resulting accuracies in terms of the number of principal components. What is striking about the resulting accuracies in Figure 5.5 is that the

Table 5.15: Performance evaluation of multi-class classification using PCA

No.Features	$PCA-RF)_{Mc-X}$			$(PCA-BN)_{Mc-X}$			$(PCA-LDA)_{Mc-X}$			$(PCA-QDA)_{Mc-X}$		
	Acc	F-M	CM_{Mc}	Acc	F-M	CM_{Mc}	Acc	F-M	CM_{Mc}	Acc	F-M	CM_{Mc}
81	0.985	0.995	0.986	0.930	0.953	0.925	0.901	0.914	0.801	0.972	0.974	0.961
70	0.996	0.988	0.988	0.948	0.964	0.942	0.894	0.906	0.735	0.967	0.975	0.967
64	0.996	0.997	0.986	0.949	0.966	0.955	0.893	0.906	0.745	0.967	0.975	0.985
59	0.996	0.995	0.987	0.952	0.967	0.917	0.893	0.906	0.677	0.967	0.975	0.880
50	0.996	0.996	0.987	0.924	0.941	0.916	0.859	0.880	0.679	0.927	0.946	0.885
40	0.996	0.997	0.9884	0.964	0.974	0.954	0.890	0.546	0.727	0.966	0.974	0.967
30	0.996	0.997	0.988	0.960	0.971	0.897	0.643	0.643	0.720	0.964	0.972	0.965
20	0.996	NAN	0.987	0.987	0.952	0.855	0.859	NAN	0.4805	0.892	0.886	0.886
10	0.996	NAN	0.986	0.953	0.964	0.946	0.850	NAN	0.363	0.856	0.886	0.886

Table 5.16: Performance evaluation of multi-class classification using AE

No.Features	$(AE-RF)_{Mc-X}$			$(AE-BN)_{Mc-X}$			$(AE-LDA)_{Mc-X}$			$(AE-QDA)_{Mc-X}$		
	Acc	F-M	CM_{Mc}	Acc	F-M	CM_{Mc}	Acc	F-M	CM_{Mc}	Acc	F-M	CM_{Mc}
81	0.985	0.995	0.983	0.930	0.953	0.895	0.912	0.922	0.908	0.968	0.969	0.920
70	0.996	0.996	0.959	0.953	0.996	0.913	0.894	0.906	0.875	0.960	0.970	0.931
64	0.996	0.996	0.985	0.955	0.968	0.954	0.900	0.908	0.743	0.960	0.969	0.963
59	0.995	0.995	0.983	0.956	0.969	0.958	0.849	0.906	0.737	0.961	0.969	0.963

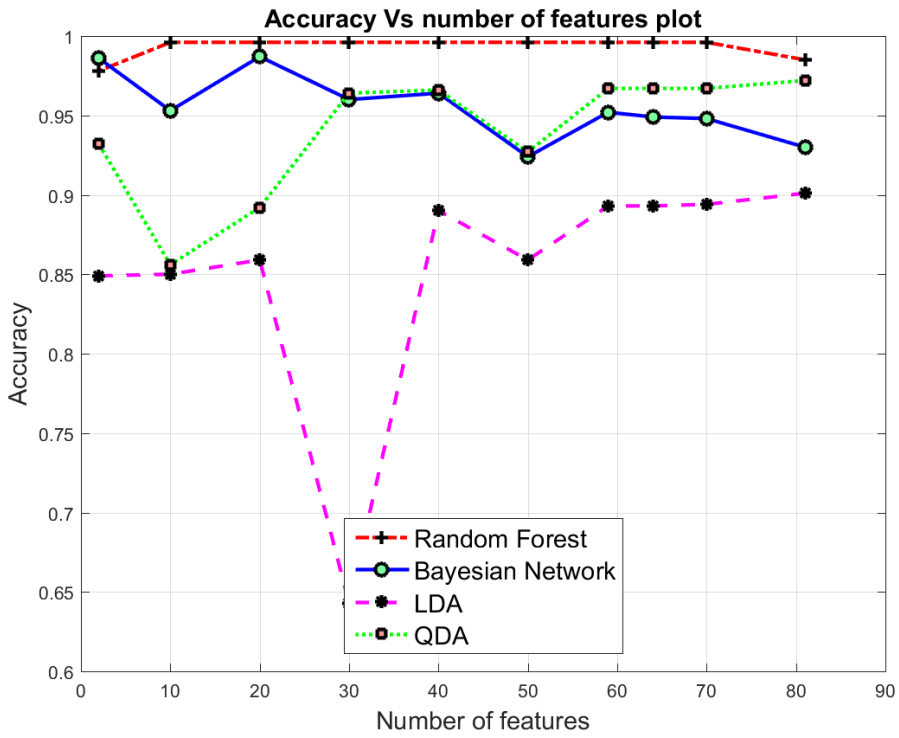

Figure 5.5: Multi-class classification: accuracy in terms of number of components using PCA

Random Forest classifier shows a constantly high accuracy for reduced features from 81 through 10. In contrast, the resulting accuracies of LDA and QDA cases were oscillatory. For QDA, the accuracy is wobbling between 66% with 10 features and 96.7% with 60 features. For LDA with 10 and 40 features, the accuracy is fluctuating between 85% and 96.6%, respectively.

The results of the AE dimensionality reduction approach are displayed in Figure 5.6. The observed accuracy for Random Forest is significant compared to LDA, QDA and the Bayesian Network classifiers. Furthermore, what stands out in this Figure 5.6, is the increase of the resulting accuracy for LDA for the reduced dimensionality from 81 through 59 features.

A detailed analysis summary of the proposed framework in terms of False Pos-

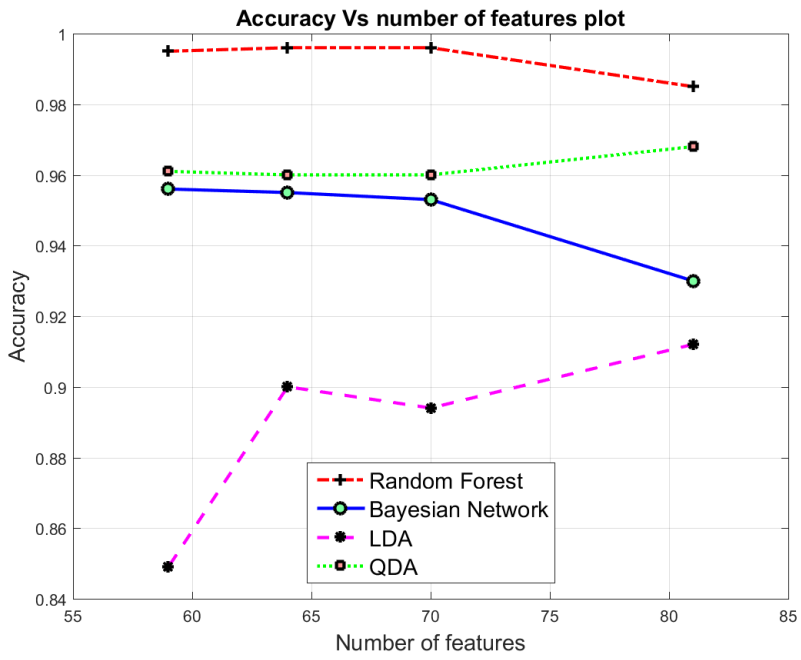

Figure 5.6: Multi-class classification: accuracy in terms of number of features using AE

itive Rate (FPR), True Positive Rate (TPR), Precision and Recall are tabulated in Tables 5.17 and 5.18. Table 5.17 depicts the results with 10 features (before applying UDBB), while Table 5.18 shows the results using 10 features (after applying UDBB). The weighted average result for all the attacks are presented in bold.

The results confirmed that the proposed framework with the reduced feature dimensionality achieved a maximum precision value of 0.996 and an FPR of 0.010, confirming the efficiency and effectiveness of the intrusion detection process. However, $(PCA - RF)_{Mc-10}$ is unable to detect the HeartBleed attacks (noted as NAN in Table 5.17). In this Table, the Recall and Precision values for HeartBleed and WebAttack:SQL are 0.00, 0.000 and 0.000, 0.000, respectively. A justification of such outcome could be due to the fact that the number of instances of HeartBleed and WebAttack:SQL originally embedded in CICIDS2017 is equal to 11 and 21, respectively. This is expected, since the total number of HeartBleed instances in the original

Table 5.17: Performance evaluation $(PCA - RF)_{Mc-10}$ before applying UDBB

	$(PCA - RF)_{Mc-10}$ Original Distribution			
	Recall	Precision	FP Rate	TP Rate
Benign	0.998	0.998	0.012	0.998
FTP-Patator	1.000	1.000	0.000	1.000
SSH-Patator	0.996	0.996	0.000	0.996
DDoS	0.877	0.900	0.001	0.877
HeartBleed	NAN	NAN	0.000	0.000
PortScan	1.000	0.998	0.000	1.000
DoSHulk	1.000	1.000	0.000	1.000
DoSGoldenEye	0.979	0.995	0.000	0.979
WebAttack: Brute Force	0.813	0.878	0.000	0.814
WebAttack:XSS	0.750	0.665	0.000	0.750
Infiltration	0.250	1.000	0.000	0.250
WebAttack:SQL	0.000	0.000	0.000	0.000
Botnet	0.960	0.991	0.000	0.960
Dos Slow HTTP Test	0.993	0.996	0.000	0.993
DoS Slow Loris	0.991	0.999	0.000	0.991
Weighted Average	**0.996**	**0.965**	**0.010**	**0.996**

Table 5.18: Performance evaluation of $(PCA - RF)_{Mc-10}$ after applying UDBB

	$(PCA - RF)_{Mc-10}$ UDBB			
	Recall	Precision	FP Rate	TP Rate
Benign	1.000	1.000	0.000	1.000
FTP-Patator	1.000	1.000	0.000	1.000
SSH-Patator	1.000	1.000	0.000	1.000
DDoS	1.000	1.000	0.000	1.000
HeartBleed	1.000	1.000	0.000	1.000
PortScan	1.000	0.999	0.000	1.000
DoSHulk	0.999	1.000	0.000	0.999
DoSGoldenEye	1.000	1.000	0.000	1.000
WebAttack: Brute Force	0.945	0.891	0.008	0.945
WebAttack:XSS	0.884	0.943	0.004	0.884
Infiltration	1.000	1.000	0.000	1.000
WebAttack:SQL	1.000	0.998	0.000	1.000
Botnet	1.000	1.000	0.000	1.000
Dos Slow HTTP Test	0.999	0.999	0.000	0.999
DoS Slow Loris	0.999	0.999	0.000	0.999
Weighted Average	**0.988**	**0.989**	**0.001**	**0.988**

dataset is 11 instances. Thus, these instances were miss-classified by the classifier. To resolve this issue and to assure that the achieved accuracy is reflected due to the effective reduction approach, this work applies the uniform distribution based balancing technique to overcome the imbalanced class distributions of certain attacks in CICIDS2017.

Table 5.19 shows the performance before and after applying the UDBB approach. As observed, $(PCA-RF)_{Mc-10}$ achieved 99.6% and 98.8% before and after applying UDBB, respectively. In the same manner, $(PCA-QDA)_{Mc-10}$ achieved 85.6% and 98.9% before and after applying UDBB, respectively. The highest achieved F-M was obtained by $(PCA-QDA)_{Mc-10}$. However, the highest $Combined_{Mc}$ metric $(CM_{(Mc)})$ achieved was 98.6% by $(PCA-RF)_{Mc-10}$.

The performance evaluation of $(PCA-X)_{Bc-10}$ and $(PCA-X)_{Mc-10}$ in terms of the time to build and test the model is presented in Table 5.20 (X represents the classifier in this table). The lowest times to test the model were achieved by LDA with 2.96 seconds for multi-class and 5.56 seconds for binary class classification.

Here, the Random Forest classifier that has the best detection performance, comes with the highest overhead in terms of the time to build and test the model. The fundamental notion behind Random Forest is that it combines many decision trees into a single model and specifically in this work, the dataset has over 2.5 million instances in total. This is expected since the worst case time complexity of Random Forest is estimated using Equation 5.12 [219].

$$O(MKN^2logN) \tag{5.12}$$

where K is the number of trees, M is the number of variables used in each split, and N is the number of training samples.

Moreover, a visualization of the dataset with two PCA components before and after applying the distribution-based balancing approach is displayed in Figures 5.7

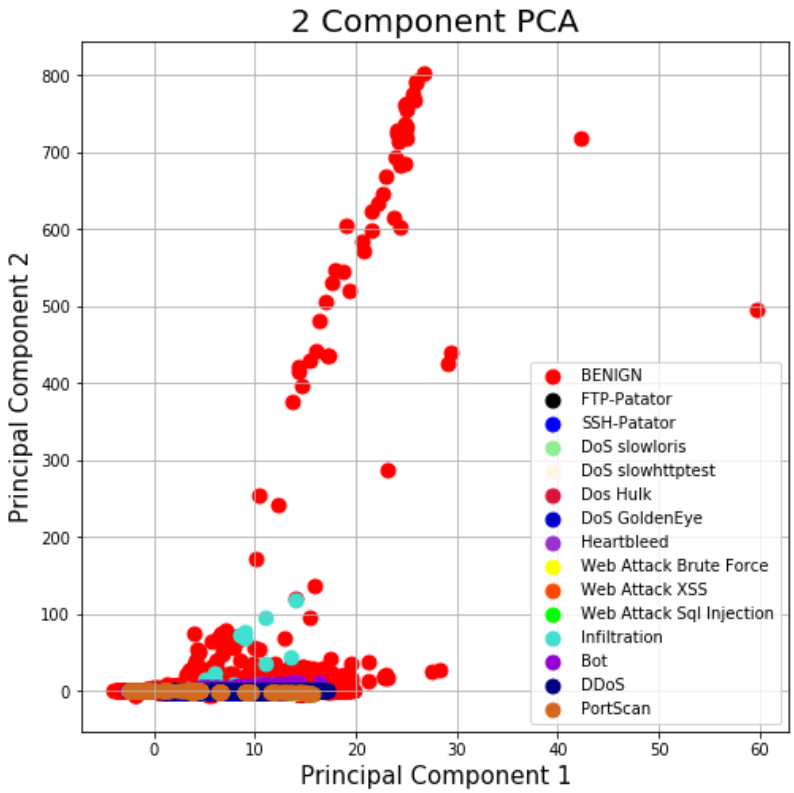

Figure 5.7: 2D Visualization of PCA on CICIDS2017 with original distribution

and 5.8, respectively. This observation of the CICIDS2017 dataset visually represents how the instances are set apart. As displayed in Figure 5.8, the same type of instances were positioned (clustered) together in groups. This shows a significant improvement over the PCA visualization before applying UDBB. Here, the normal instances are

Table 5.19: Performance evaluation of $(PCA-X)_{Mc-10}$

Classifier	Acc	F-M	$CM_{(Mc)}$
Before applying UDBB			
PCA-Random Forest $(PCA-RF)_{Mc-10}$	0.996	NAN	0.9866
PCA-Bayesian Network $(PCA-BN)_{Mc-10}$	0.953	0.964	0.9464
PCA-LDA $(PCA-LDA)_{Mc-10}$	0.850	NAN	0.3626
PCA-QDA $(PCA-QDA)_{Mc-10}$	0.856	0.886	0.8862
After applying UDBB			
PCA-Random Forest $(PCA-RF)_{Mc-10}$	0.988	0.988	0.9882
PCA-Bayesian Network $(PCA-BN)_{Mc-10}$	0.976	0.977	0.9839
PCA-LDA $(PCA-LDA)_{Mc-10}$	0.957	0.957	0.6791
PCA-QDA $(PCA-QDA)_{Mc-10}$	0.989	0.990	0.8851

Table 5.20: Time to build and test the models

Classifier	Time to Build the Model (Sec.)	Time to Test the Model (Sec.)
Binary-class Classification		
LDA	12.16	5.56
QDA	12.84	6.57
RF	752.67	21.52
BN	199.17	11.07
Multi-class Classification		
LDA	17.5	2.96
QDA	15.35	3.16
RF	502.81	41.66
BN	175.17	10.07

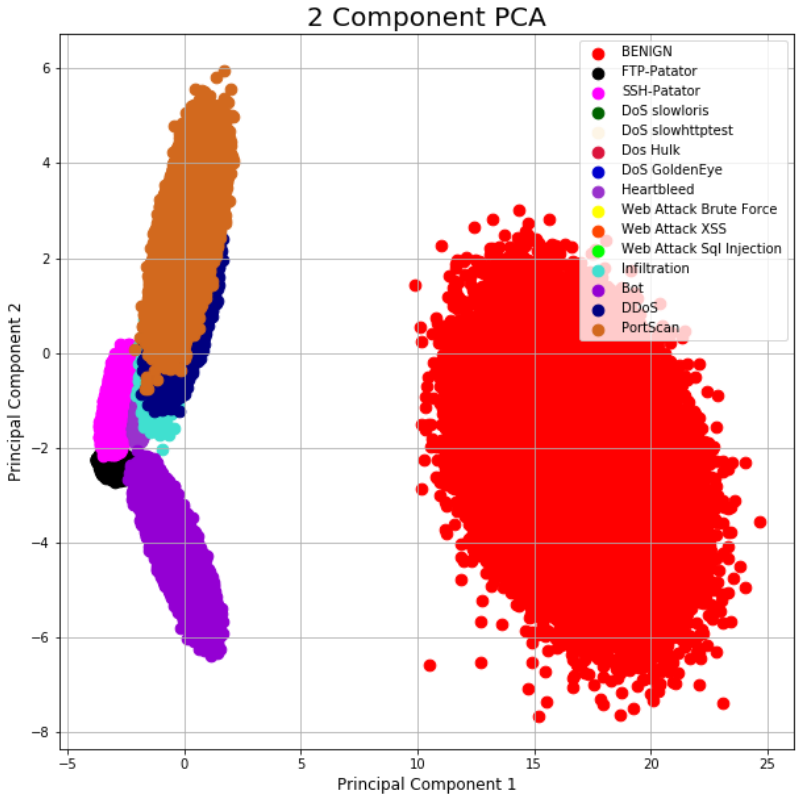

Figure 5.8: 2D Visualization of PCA on CICIDS2017 with UDBB

very clearly clustered in their own group. This is applied for other types of instances as well.

The confusion matrix for the $PCA - RF_{Mc-10}$ is shown in Figure 5.9.

The value for HeartBleed is reported as NAN (Not A Number). These values result from operations which have undefined numerical values. The classifier $PCA - RF_{Mc-10}$ fails to classify HeartBleed attacks. In contrast, as a result of applying the UDBB technique, the $PCA - RF_{Mc-10}$ is able to detect 100% of HeartBleed attacks, as indicated from the confusion matrix in Figure 5.10.

As exemplified from the obtained results, the PCA approach is able to preserve important information in CICIDS2017, while efficiently reducing the features dimensions in the used dataset, as well as it also presents a reasonable visualization model of the data.

Features such as Subflow Fwd Bytes, Flow Duration, Flow Inter arrival time (IAT), PSH Flag Count, SYN Flag Count, Average Packet Size, Total Len Fwd Pck, Active Mean and Min, ACK Flag Count, and Init_Win_bytes_fwd are observed to be the discriminating features embedded in CICIDS2017 [8].

5.4.3 Performance Comparison - CICIDS2017

A comparison between the proposed framework and related work is highlighted in Table 5.21. The authors in [116], [120], and [121] reported the accuracy.While others reported the Accuracy,since the CICIDS2017 dataset is imbalanced Dataset we used F- Measure to compare and evaluate our frame work.As illustrated in Table 5.21 Our proposed framework outperforms previous studies in terms of F-Measure as well as accuracy.

Table 5.21: Comparison of CICIDS2017's related studies and performances

Reference	Classifier name	F-measure	Feature selection/extraction (Features Count)
[8]	KNN RF ID3 Adaboost MLP Naïve Bayes QDA	0.96 0.97 0.98 0.77 0.76 0.04 0.92	Random Forest Regressor (54)
[112]	MLP	0.948	Payload related features
[114]	SVM	0.921	DBN
[113]	KNN	0.997	Fisher Scoring (30)
[119]	XGBoost for DoS Attacks	0.995	(80)
[120]	Deep Learning for Port Scan Attacks	Accuracy 97.80	(80)
[120]	SVM for Port Scan Attacks	Accuracy 69.79	(80)
[116]	XGBoost	Accuracy 98.93	DDR Features Selections (36)
[121]	Deep Multi Layer Perceptron (DMLP) for DDoS Attacks	Accuracy 91.00	Recursive feature elimination with Random Forest
Proposed Framework	**Random Forest**	**0.995**	**Auto-encoder (59)**
Proposed Framework	**Random Forest**	**0.996**	**PCA with Original Distribution (10)**
Proposed Framework	**Random Forest**	**0.988**	**PCA With UDBB(10)**

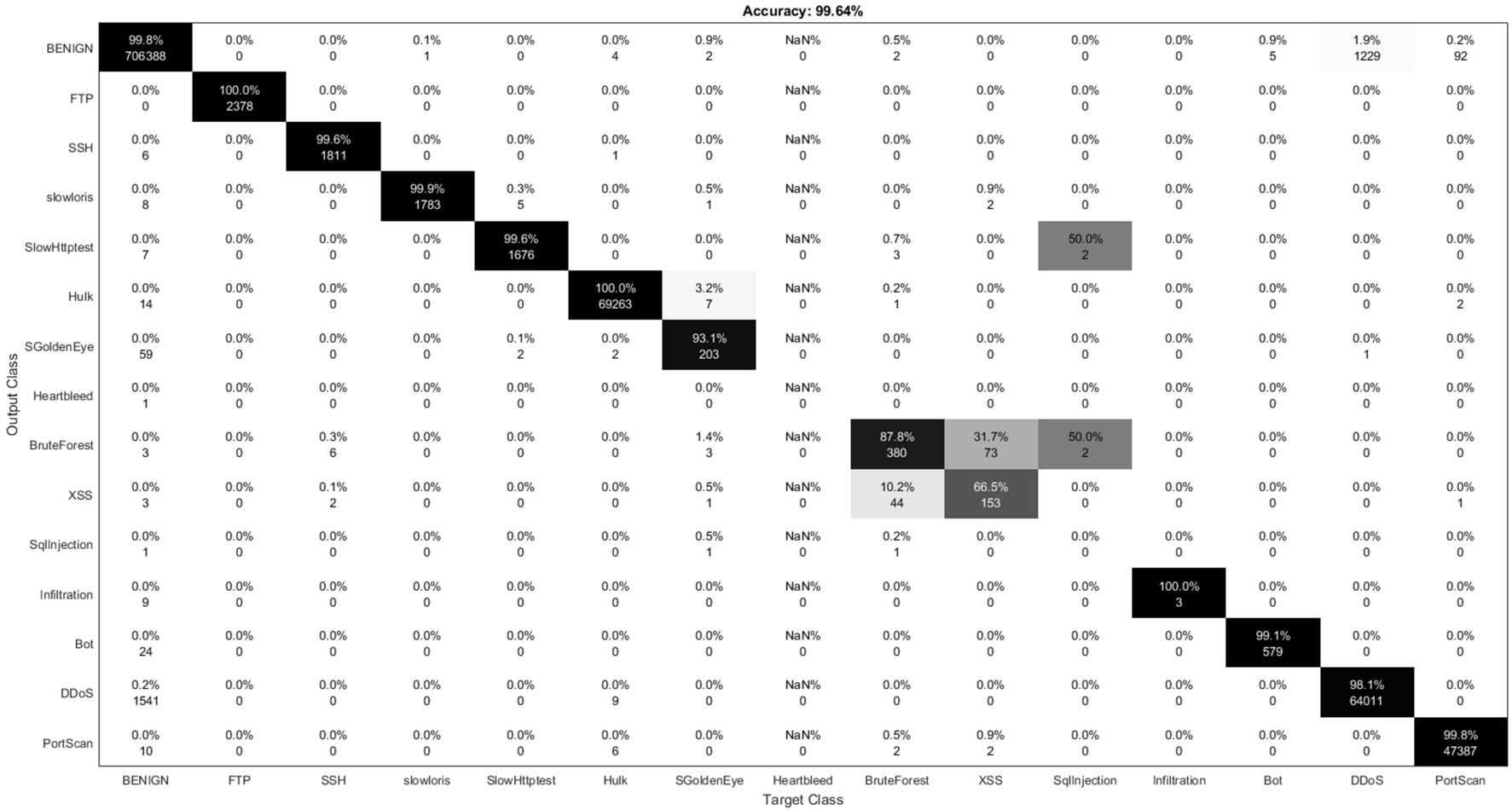

Figure 5.9: Confusion matrix for $(PCA - RF)_{Mc-10}$ with original class distribution

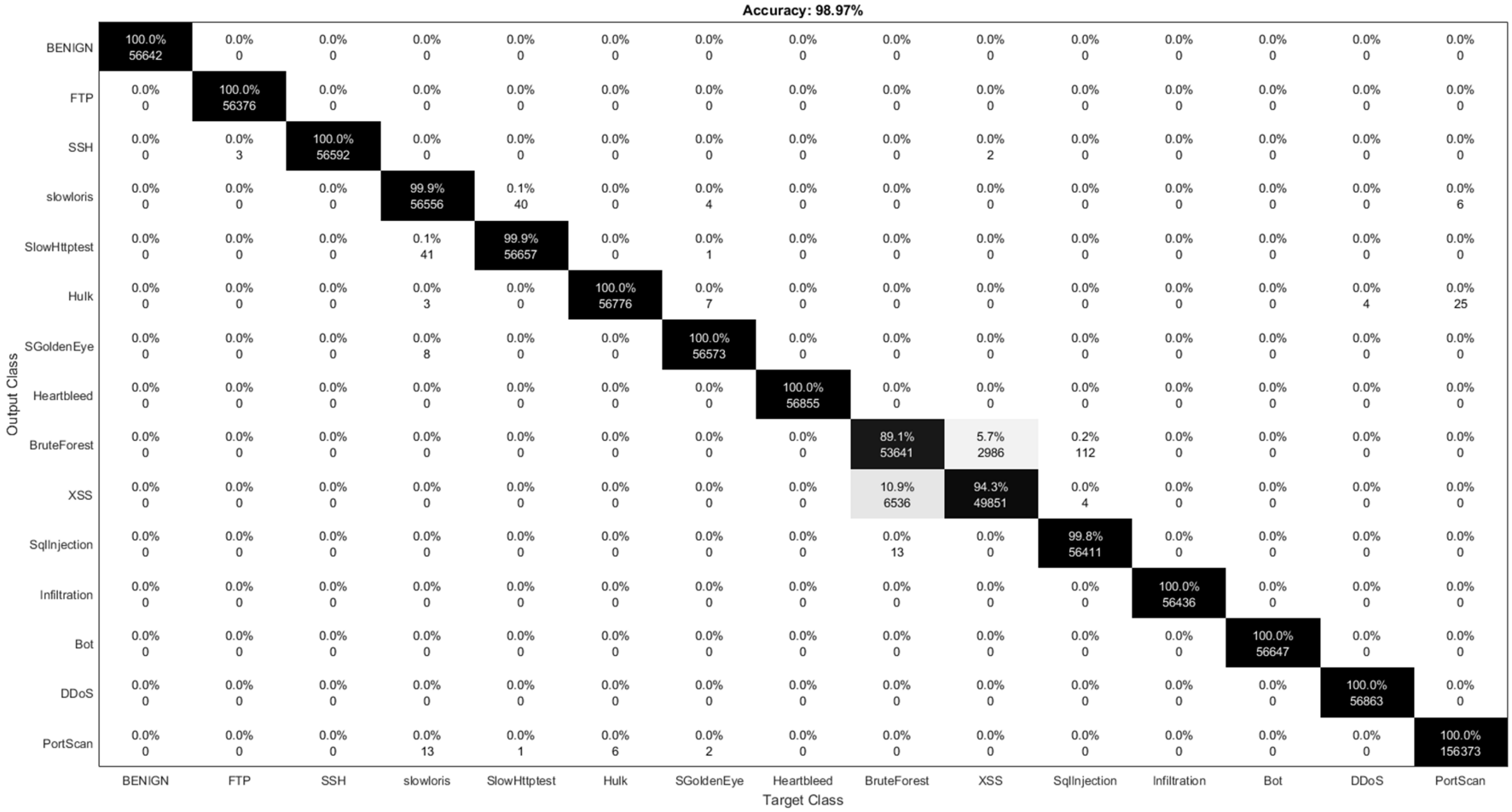

Figure 5.10: Confusion matrix for $(PCA - RF)_{Mc-10}$ with UDBB

5.4.4 Results and Discussion CIDDS-001

When using the CIDDS-001 dataset, the experiments and simulations were carried out using an Intel Core i7 with 3.30 GHz and 32 GB RAM running Windows 10 and the Waikato Environment for Knowledge Analysis (Weka) software along with MATLAB toolset. Additionally, Keras, an open source neural network library along with the synthetic minority reconstruction technique (SMRT), was deployed for the variational autoencoder (VAE).

This research utilizes the following machine learning classification approaches for intrusion detection in binary classification on the CIDDS-001 benchmark dataset:

- Deep Neural Networks (DNN) [177] with three different architectures: DNN (3,512,0.1), DNN (3,1024,0.1) and DNN (3,2048,0.1). Each architecture has three layers and a learning rate of 0.1. The size of the middle layer is 512, 1024, and 2048 neurons, respectively.
- Random Forest (RF) [180]
- Voting technique [93] to vote on OneR, Naïve Bayes, and ExtraTree
- Stacking technique [183] with Linear discriminant analysis (LDA), Naïve Bayes and OneR
- Variational Auto-Encoder (VAE) [177] with original class distribution (Table 5.22 introduces the VAE's architectural properties and settings)

The experiments are carried out under two scenarios: With Handling Imbalanced Class Distribution (WHICD), and Without Handling Imbalanced Class Distribution (WoHICD). See Tables 5.23, 5.24 and 5.25. In the WHICD scenario, we apply sampling techniques to handle the imbalanced class distribution of CIDDS-001 before feeding the data to the classifiers. In the WoHICD scenario, we apply the machine

Table 5.22: Variational Auto-Encoder properties

Parameter	Value	Parameter	Value
Activation function	Elu	Optimizer	Adam
μ	0	Hidden Size	7
δ	1	Latent dimension	2
No. of Epoch	75	Input Layer	10
Batch Size	128	Learning Rate	0.001

Table 5.23: Deep learning performance evaluation of WHICD and WoHICD scenarios in terms of Accuracy

	Accuracy (%)				
	Original Distribution	Down-Sampling	Up-Sampling	Class Balancer	Spread Sub-Sample
DNN (3,512,0.1)	98.71	98.06	72.06	99.71	94.59
DNN (3,1024,0.1)	99.25	99.01	89.84	98.24	97.26
DNN (3,2048,0.1)	99.65	99.30	94.27	98.77	96.80

Table 5.24: Performance evaluation of WHICD and WoHICD scenarios in terms of precision, recall, and G-Mean metrics

	Original Distribution			Down-Sample			Up-Sample			Class Balancer			Spread Sub-Sample		
	Pre.	Rec.	G-M	Pre.	Rec.	G-M	Pre.	Rec.	G-M	Pre	Rec.	G-M	Pre.	Rec.	G-M
Random Forest	1.00	0.999	0.999	1.00	1.00	1.00	0.994	0.994	0.994	1.00	1.00	1.00	0.993	0.993	0.993
Voting	1.00	1.00	1.00	1.00	1.00	1.00	0.990	0.983	0.986	0.997	0.997	0.997	0.990	0.980	0.984
Stacking	0.997	0.997	0.997	0.996	0.996	0.996	0.998	0.998	0.998	0.997	0.997	0.997	0.995	0.994	0.994

Table 5.25: Performance evaluation of WHICD and WoHICD scenarios in terms of Acc, DR, FAR, and combined metrics (Comb.)

	Original Distribution				Down-Sampling				Up-Sampling				Class Balancer				Spread Sub-Sample			
	Acc.	DR	FAR	Comb.	Acc.	DR	FAR	Comb.	Acc.	DR	FAR	Comb.	Acc.	DR	FAR	Comb.	Acc.	DR	FAR	Comb.
Random Forest	0.999	0.999	1E-4	0.999	0.999	0.999	1E-4	0.999	0.993	0.723	0.0006	0.993	0.999	0.999	1E-4	0.999	0.987	0.724	0.0011	0.857
Voting	0.999	0.990	0.010	0.984	0.999	0.990	8E-4	0.994	0.989	0.999	0.0101	0.984	0.996	0.994	7E-4	0.995	0.980	0.999	0.0203	0.969
Stacking	0.996	0.906	0.001	0.998	0.995	0.910	0.0025	0.950	0.998	0.984	0.0016	0.998	0.996	0.909	0.469	0.483	0.993	0.994	0.0063	0.987

learning classifiers to the original class distribution. We compare and contrast the two scenarios for the classifiers.

What stands out from these Tables is that the accuracy for the original distribution was 99.65% for DNN (3,2048,0.1). In the same manner, with down-sampling, the highest accuracy was 99.30% for DNN (3,2048,0.1), whereas the lowest accuracy was 72.06% using DNN (3,512,0.1). Surprisingly, the same classifier, with the same architecture, DNN (3,512,0.1) achieved 99.30% accuracy for the up-sampling approach. In case of Random Forest, the highest achieved accuracy was 99.99% with the original

distribution, down-sampling and class balancer, respectively. Moreover, with voting, the accuracy was 99.99% with both the original distribution as well as down-sampling. In Stacking, the highest accuracy was 99.80% with up-sampling. Lastly, for the VAE, the accuracy was 97.59% with the original distribution. The VAE architecture of (10,7,2) was applied for the encoder and (2,7,10) for the decoder, with a learning rate of 0.001, and ELU activation function (Table 5.22).

To sum up, RF's performance was better for the original distribution, down-sampling, and class balancer. Both the RF and DNN approaches have many advantages that make these two classifiers outperform others. More specifically, the Random Forest classifier has the ability to select the most discriminatory features and is able to deal with noisy or incomplete data. The deep learning approach is likely to outperform other approaches when there is large amount of data. Moreover, the deep learning approach is capable of taking into account a wide variety of features. In our experiments, the Random Forest classifier outperformed deep learning and stacking techniques. In up-sampling, the generated data might contain noisy or incomplete data and the Random Forest classifier has the ability to deal with these issues. In spread sub-sample, the size of the data was reduced. It is known that the size of the data has an effect on the performance of the deep learning approach. Conversely, the size of data has less effect on the performance of Random Forest. Thus, Random Forest outperformed deep learning in the spread sub-sample technique.

5.4.5 Performance Comparison - CIDDS-001

Finally, a comparison with previous work is highlighted in Table 5.26. The achieved results for Voting-WHICH, RF-WHICD, and Stacking-WHICH outperformed previous research work in terms of accuracy.

Table 5.26: Comparison with prior CIDDS-001 related work

Study	Approach	Accuracy (%)
Verma and Ranga [128]	2NN	99.60
Verma and Ranga [50]	DT	99.90
Tama and Rhee [123]	DNN-10-FCV	99.90
Idhammad et al. [125]	Entropy+RF	99.54
Proposed Approach-1	DNN-WHICD	99.71
Proposed Approach-2	Voting-WHICD	99.99
Proposed Approach-3	RF-WHICD	99.99
Proposed Approach-4	Stacking-WHICD	99.80

5.5 MCPS IDS Co-Design

In this dissertation, a behavior monitoring tool for four medical devices that include PCAg, CD, VSM, and CGM devices, respectively were developed. We used Mentor Graphics ModelSim-Altera software and Quartus II software to simulate and synthesize the design of the idealized finite state machine (FSM) hardware model that reflect a Behavior Specification Rules Monitoring (BSRM) tool. Timing simulation was applied to the design files to make sure that the logic of the design is correct and to ensure that the synthesized design meets the functional and timing requirements and behaves as expected. In addition, software-based machine learning approach were also applied to the same medical CPS IDS.

5.5.1 Results and Discussion of Simulation and Hardware Synthesis for MCPS IDS

Tables 5.27 through 5.34 the show resource utilization and power analysis summaries for PCAg, VSM, CD, and CGM, respectively. Furthermore, Tables 5.35 through 5.38 show timing analysis summaries for PCAg, CD, VSM, and CGM, respectively.

Through our simulation and synthesis, we demonstrated that the BSRM tool

Table 5.27: Utilization summary for PCAg BSRM

Attribute	Criteria
Family	Cyclone III
Device	EP3C120F780C7
Timing Models	Final
Total logic elements	57 / 119,088 (<1 %)
Total combinational functions	57 / 119,088 (<1 %)
Dedicated logic registers	4 / 119,088 (<1 %)
Total registers	4
Total pins	32 / 532 (6 %)

Table 5.28: Power analysis summary for PCAg BSRM

Attribute	Criteria
Family	Cyclone III
Device	EP3C120F780C7
Power Models	Final
Total Thermal Power Dissipation	120.48 mW
Core Dynamic Thermal Power Dissipation	1.64 mW
Core Static Thermal Power Dissipation	99.06 mW
I/O Thermal Power Dissipation	19.77 mW

Table 5.29: Utilization summary for VSM BSRM

Attribute	Criteria
Family	Cyclone III
Device	EP3C120F780I7
Timing Models	Final
Total logic elements	56 / 119,088 (<1 %)
Total combinational functions	56 / 119,088 (<1 %)
Dedicated logic registers	4 / 119,088 (<1 %)
Total registers	4
Total pins	45 / 532 (8 %)

can effectively identify the expected normal behavior of the device and detect any deviation from its normal behavior. Furthermore, the model is consistent with the requirements for lower power consumption and higher bandwidth applications. The FPGA module of the BSRM can be embedded in medical devices in order to detect any

Table 5.30: Power analysis summary for VSM BSRM

Attribute	Criteria
Family	Cyclone III
Device	EP3C120F780I7
Power Models	Final
Total Thermal Power Dissipation	121.77 mW
Core Dynamic Thermal Power Dissipation	1.67 mW
Core Static Thermal Power Dissipation	99.06 mW
I/O Thermal Power Dissipation	21.04 mW

Table 5.31: Utilization summary for CD BSRM

Attribute	Criteria
Family	Cyclone IV GX
Device	EP4CGX150DF31I7
Timing Models	Final
Total logic elements	54 / 149,760 (<1 %)
Total combinational functions	54 / 149,760 (<1 %)
Dedicated logic registers	4 / 149,760 (<1 %)
Total registers	4
Total pins	36 / 508 (7 %)

Table 5.32: Power analysis summary for CD BSRM

Attribute	Criteria
Family	Cyclone IV GX
Device	EP4CGX150DF31I7
Power Models	Final
Total Thermal Power Dissipation	141.18 mW
Core Dynamic Thermal Power Dissipation	2.14 mW
Core Static Thermal Power Dissipation	118.71 mW
I/O Thermal Power Dissipation	20.33 mW

deviation from normal behavior specification. The reconfigurable FPGA chip adopts any design model according to the requirements of the device, patient, treatment algorithm, and/or pervasive healthcare application.

Table 5.33: Utilization summary for CGM BSRM

Attribute	Criteria
Family	Cyclone III
Device	EP3C120F780I7
Timing Models	Final
Total logic elements	56 / 119,088 (<1 %)
Total combinational functions	56 / 119,088 (<1 %)
Dedicated logic registers	4 / 119,088 (<1 %)
Total registers	4
Total pins	45 / 532 (8 %)

Table 5.34: Power analysis summary for CGM BSRM

Attribute	Criteria
Family	Cyclone IV GX
Device	EP4CGX110DF31I7
Power Models	Final
Total Thermal Power Dissipation	156.92 mW
Core Dynamic Thermal Power Dissipation	2.49 mW
Core Static Thermal Power Dissipation	118.74 mW
I/O Thermal Power Dissipation	35.69 mW

Table 5.35: Timing summary for PCAg BSRM

Attribute	Timing Summary
Minimum period	3.25 ns
Minimum input arrival time before clock	7.006 ns
Maximum output required time after clock	4.114 ns
Maximum combinational path delay	8.275 ns
Maximum Frequency	380.700 MHz

Table 5.36: Timing summary for VSM BSRM

Attribute	Timing Summary
Minimum period	3.025 ns
Minimum input arrival time before clock	12.601 ns
Maximum output required time after clock	5.278 ns
Maximum combinational path delay	6.256 ns
Maximum Frequency	330.600 MHz

Table 5.37: Timing summary for CD BSRM

Attribute	Timing Summary
Minimum period	2.907 ns
Minimum input arrival time before clock	14.113 ns
Maximum output required time after clock	7.645 ns
Maximum combinational path delay	16.110 ns
Maximum Frequency	343.991 MHz

Table 5.38: Timing summary for CGM BSRM

Attribute	Timing Summary
Minimum period	1.914 ns
Minimum input arrival time before clock	10.819 ns
Maximum output required time after clock	5.278 ns
Maximum combinational path delay	12.857 ns
Maximum Frequency	522.371 MHz

5.5.2 Results and Discussion of Software Machine Learning Approaches for MCPS IDS

In this dissertation, a software-based machine learning approach is adopted to compare and contrast with the hardware-based behavior specification rules co-designs for the medical CPS IDS case study. Two classifiers are used, which include Random Forest and OneR. The Random Forest classifier outperforms OneR with an overall accuracy of 98.26%. OneR resulted in a lower accuracy of 70.28%.

Tables 5.39 and 5.40 provide the summary statistics of the performance evaluation metrics for the CGM's software-based machine learning behavior monitoring tool using Random forest and OneR, respectively. The lowest achieved weighted average FP rate is 0.014 using Random Forest, whereas the achieved weighted average FP rate is 0.250 utilizing OneRr. Moreover, the precision value is 0.983 and 0.708 with Random Forest and OneR, respectively. The time to build and test the model are exemplified in Table 5.41.

Finally, the confusion matrix charts, Figures 5.12 and 5.11, show the perfor-

Table 5.39: CGM's machine learning results with Random Forest

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.953	0.000	1.000	0.953	0.976	0.974	0.999	0.993	Warning
1.000	0.022	0.970	1.000	0.985	0.974	1.000	1.000	Safe
0.973	0.008	0.991	0.973	0.982	0.965	0.999	0.999	Unsafe
0.983	0.014	0.983	0.983	0.983	0.970	1.000	0.999	Weighted Avg.

Table 5.40: CGM's machine learning results with OneR

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.163	0.005	0.778	0.163	0.269	0.332	0.579	0.205	Warning
0.689	0.220	0.693	0.639	0.691	0.469	0.734	0.608	Safe
0.818	0.322	0.708	0.708	0.759	0.500	0.748	0.668	Unsafe
0.703	0.250	0.708	0.708	0.685	0.471	0.727	0.599	Weighted Avg.

Table 5.41: Time to build and test the models

Classifier	Time to Build the Model (Sec.)	Time to Test the Model (Sec.)
Random Forest	0.5	0.6
OneR	0.02	0.01

mance analysis of classification models for Random Forest and OneR.

The findings demonstrate that for this medical CPS IDS case study, the hardware-based behavior rules approach achieve better performance metrics results for determining the malicious behavior of the embedded medical devices, as the safe and malicious behaviors can be defined as specified deterministic rules.

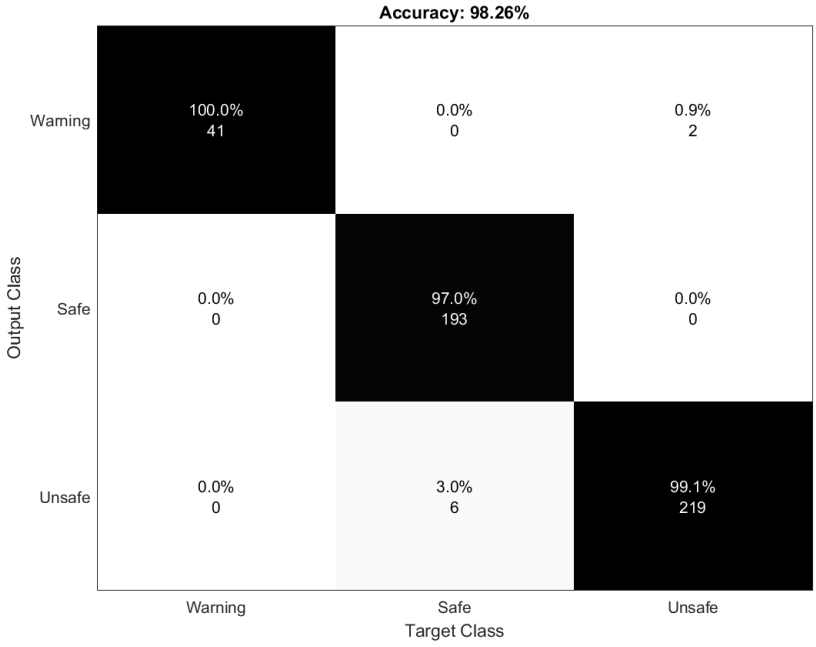

Figure 5.11: Random Forest confusion matrix

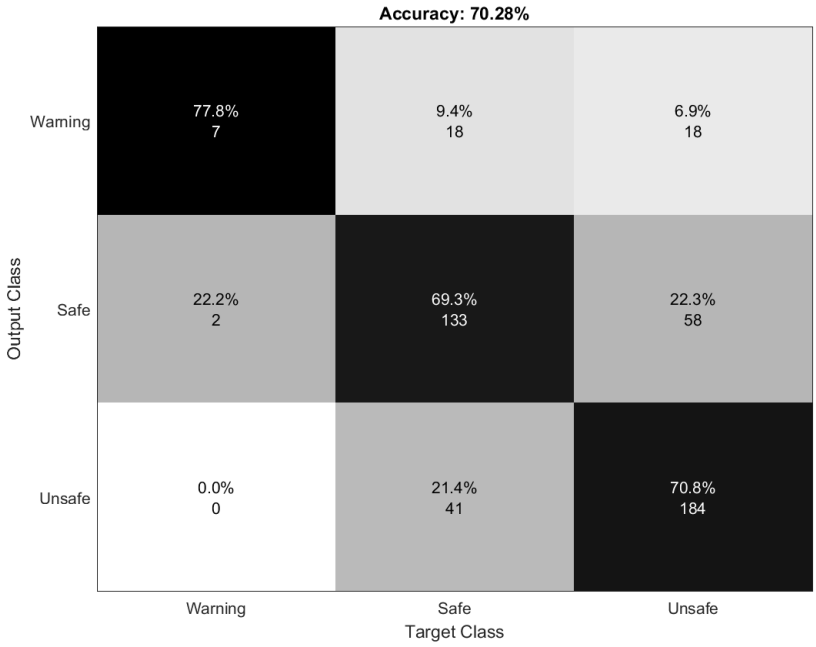

Figure 5.12: OneR confusion matrix

CHAPTER 6: STATISTICAL ANALYSIS

6.1 Statistical Power Analysis

In this section, statistical power analysis [220],[221] is performed to estimate and ensure the appropriate sample size (e.g. the minimum number of instances that need to be utilized in this study). Figure 6.1 depicts the general steps for carrying out statistical power analysis [221].

Experimental results show that the accuracy of the IDS using Random Forest with the proposed five features (5-FSG) outperform the case with 32 features (FSG-32). Therefore, the power analysis is also used to accept this hypothesis that is stated

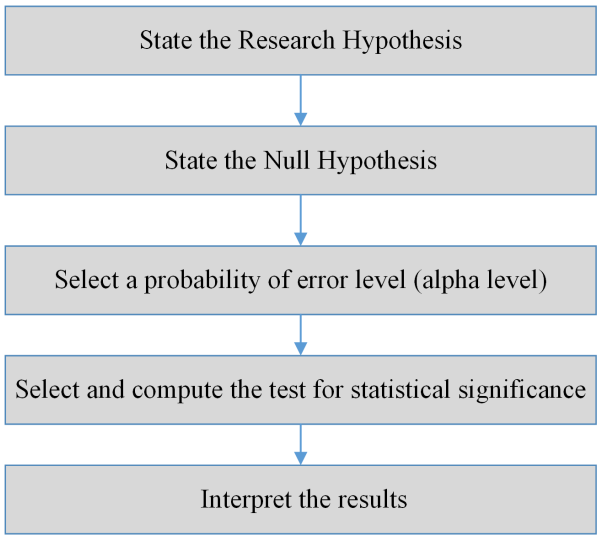

Figure 6.1: Statistical power analysis steps

as an alternative hypothesis H_a, whereas the null hypothesis H_0 is the hypothesis where there is no change in the accuracy using the proposed features with respect to the 32 features (FSG-32). With the power analysis, a statistical test rejects the null hypothesis when it is false. Here, the conclusion is that there is a difference between the accuracies (better accuracy) using 5 features (FSG-5) and we can accept our alternative hypothesis H_a. If the null hypothesis is not rejected, then the alternative hypothesis should be rejected. The opposing hypotheses for our work can be stated as follows:

$$H_0 : \mu_{5-FSG} = \mu_{original} \tag{6.1}$$

$$H_a : \mu_{5-FSG} > \mu_{32-FSG} \tag{6.2}$$

where μ_{FGS-5} is the average accuracy of Random Forest using the 5 selected features and μ_{FGS-32} is the accuracy average of Random Forest using the 32 features for classification.

To determine the required sample size n, four parameters/factors must be known or estimated:

- α : Significance level (1% or 5%)
- p : Desired power of the test (80%)
- σ : Population standard deviation.
- d : Effect size (the difference between the two groups)

The values of the first two parameters are generally fixed. The parameter of significance level α is usually set to either 0.05 or 0.01 and is the probability of rejecting the null hypothesis when it is true. The power parameter p is the probability that the effect will be detected and is usually set to either 0.8 or 0.9. On the other

hand, the last two parameters are problem dependent. For our analysis, the last two parameters are estimated based on our experiments and results. Thus, the values of all the four required parameters are set as below:

- $\alpha = 5\%$
- $p = 80\%$
- $\sigma = 0.217794307$
- $d = 0.0029$

Using these parameters together with the z-test model to obtain z-scores, the sample size n can be computed by using Equation 6.3.

$$Sample\ size\ (n) = 2 \times \left(\sigma \times \frac{z_{1-\frac{\alpha}{2}} + z_p}{d}\right)^2 \tag{6.3}$$

Given the estimated values of the required parameters, we will have:

$$Sample\ size\ (n) = 2 \times \left(0.218 \times \frac{z_{1-\frac{0.05}{2}} + z_{0.8}}{0.0029}\right)^2$$

$$Sample\ size\ (n) = 2 \times \left(0.218 \times \frac{1.959 + 0.8416}{0.0029}\right)^2$$

$$Sample\ size\ (n) = 88,644$$

Using the obtained sample size n and the significance level α, the below parameters can be computed in order to apply the z-test and then make a decision on accepting or rejecting our alternative hypothesis:

$$Mean\ (\hat{x}) = \frac{\sum x}{n} = \frac{82970}{88644} = 0.936 \tag{6.4}$$

$$Variance\ (\sigma^2) = \frac{\sum (x - \hat{x})^2}{n} = 0.01959536 \tag{6.5}$$

$$Standard\ Deviation\ (\sigma) = \sqrt{\sigma^2} = 0.1399834276 \tag{6.6}$$

$$Critical\ z = z_{1-\frac{\alpha}{2}} = 1.96 \tag{6.7}$$

$$Standard\ Error\ (Sx) = \frac{\sigma}{\sqrt{n}} = 4.701667912E-4 \tag{6.8}$$

$$Lower\ limit = \hat{x} - (Critical\ z \times Sx) = 0.935 \tag{6.9}$$

$$Upper\ limit = \hat{x} + (Critical\ z \times Sx) = 0.937 \tag{6.10}$$

$$Null\ Hypothesis\ (H_0) : \mu_{FGS-5} = \mu_{FGS-32} = 0.915 \tag{6.11}$$

$$Z\_test\ (Z) = \frac{\hat{x} - \mu_{FGS-5}}{Sx} = 45 \tag{6.12}$$

Since the obtained value of the z-test is higher than the critical value ($45 > 1.96$), the observed difference is significant and shows that the selected 5 features enhance the accuracy of Random Forest for IDS. In other words, the results of the z-test show that the null hypothesis (H_0) should be rejected, and the sample set of $88,644$ instances is sufficient to prove that Random Forest with the 5-FSG is more accurate than Random forest with 32-FSG. Our experimental results included over 575,643 instances which is more than sufficient to prove the hypothesis claims.

6.2 Wilcoxon Signed-Ranks Test

This section performs a Wilcoxon Signed-Ranks Test (WSRT) [222], [223] analysis to estimate and rank the module of the performance differences between the

different algorithms that were utilized in this dissertation.

To be more specific, our target is to test two algorithms in several datasets. The Wilcoxon Signed Rank Test (WSRT) is a non-parametric test and is used when one or more of the datasets are not normally distributed. Furthermore, it ranks the differences in performances of two classifiers for each dataset, ignoring the signs, and compares the ranks for the positive and the negative differences.

Experimental results show that the there is a significance difference in the achieved accuracy using Random Forest in different datasets compared to the achieved accuracy using Bayesian Network in the same dataset. Therefore, the WSRT analysis is also used to prove this hypothesis that is stated as an alternative hypothesis H_1, whereas the null hypothesis H_0 is the hypothesis where there is no significant difference in the achieved accuracy using these two classifier models in different dataset.

With the WSRT analysis, a statistical test is rejected (the null hypothesis is false) when we get a test statistically less than or equal to the critical value [222]. Here, the conclusion is that there is a difference between the accuracies (better accuracy) using Random Forest, thus can confirm our alternative hypothesis H_1. If the null hypothesis is not rejected (the null hypothesis is true), then the alternative hypothesis should be rejected. The opposing hypotheses for our work in this analysis can be stated as follows:

$$H_0 : Acc_{RandomForest} = Acc_{BaesianNetwork} \tag{6.13}$$

$$H_1 : Acc_{RandomForest} \neq Acc_{BaesianNetwork} \tag{6.14}$$

Using the obtained accuracies of both classifiers, the number of datasets N and the significance level α, the below parameters can be computed in order to apply the WSRT analysis and then make a decision on accepting or rejecting our alternative hypothesis. The WSRT is described as follows.

• Calculate R^+ as the sum of ranks for the datasets on which the Random Forest algorithm outperformed the Bayesian Network.

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{6.15}$$

where d_i is the difference between the performance scores of the two classifiers on i out of N datasets.

• Calculate R^- as the sum of ranks for the opposite case; the datasets on which the Bayesian Network algorithm outperformed the Random Forest classifier.

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \tag{6.16}$$

Again, where d_i is the difference between the performance scores of the two classifiers on i out of N data sets.

• Estimate T as the smaller of the sums, $T = min(R^+, R^-)$

• Set the confidence level of α =0.20

• Set the number of datasets to 9

In our experiments, the procedure for WSRT is illustrated in Table 6.1 below:

Table 6.1: Wilcoxon signed-ranks test analysis for multi-class classification

No. of Features	Dataset	Random Forest	Bayesian Network	Differences	Absolute Difference	Rank
10	CICIDS2017	0.996	0.953	- 0.043	0.043	7
10	CICIDS2017-UDBB	0.988	0.976	-0.012	0.012	4
10	CIDDS-001	0.999	0.998	- 0.001	0.001	1
10	CIDDS-001-RWR	0.993	0.999	0.006	0.006	5
10	CIDDS-001-RWoR	0.999	0.999	0.000	0.000	–
5	AWID	0.994	0.998	0.004	0.004	3
5	AWID-SMOTE	0.993	0.963	- 0.030	0.030	5
5	AWID-RWR	0.994	0.963	- 0.031	0.031	6
5	AWID-RWoR	0.984	0.964	-0.002	0.002	2

Table 6.1 shows the comparison of Random Forest and Bayesian Network accuracies. The experiments were performed on 9 datasets.

We are trying to reject the null-hypothesis H_0 which sttaes that both algorithms perform equally well. The ranks are assigned from the lowest to the highest absolute

difference.

The sum of ranks for the positive differences is $R^- = 1+2+4+5+6+7 = 25$. The sum of ranks for the negative differences is $R^+ = 3+5 = 8$. From the calculations, $WSRT_{stat} = T = min(R^-, R^+) = min(25, 8) = 8$, and $WSRT_{Critical} = 10$.

If $WSRT_{stat} < WSRT_{Critical}$, we reject the null hypothesis. According to the table of exact critical values for the WRST, for a confidence level of $\alpha = 0.2$ and N=9 datasets, the difference between the classifiers is significant if the smaller of the sums is equal or less than 10. We therefore reject the null hypothesis.

CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

There are a number of unexplored areas in which future work could be carried out to advance upon what has been achieved in the context of this dissertation. In the following, we elaborate on these possible future directions.

7.1 Intrusion Detection: Software-based Machine Learning

7.1.1 AWID

One of the most challenging problems in the area of machine learning is feature selection since the selected attributes (features) have an impact on the classification results. Twelve well known classifier algorithms; which are J48, OneR, ZeroR, Naïve Bayes, Random Forest, Simple Logistic, Bagging, LogitBoost, AdaBoost, Random Tree, Multi-Layer Perceptron, and Multi-Layer Perceptron-CS; were evaluated through four different attribute sets of 32, 10, 7 and 5 FSGs. The results confirm that optimum attribute selection/reduction can lead to better results in terms of accuracy and processing time.

7.1.2 CICIDS2017

The dissertation examined incorporating auto-encoder and PCA for dimensionality reduction and the use of classifiers towards designing an efficient intrusion

detection system on the CICIDS2017 dataset. The experimental analysis confirmed that the adopted feature dimensionality reduction technique leads to better results in terms of several performance metrics as well classification speeds. These findings highlight the potential usefulness of auto-encoder and PCA in dimensionality reduction for IDS. Moreover, the AE and PCA are both feasible and effective dimensionality reduction techniques for designing efficient intrusion detection systems.

The large number of decision trees that the Random Forest classifier produced by randomly selecting a subset of training samples and a subset of variables for splitting at each tree node, makes the Random Forest classifier less sensitive to both the quality of training instances as well as the over fitting issue. Moreover, Random Forest is suitable, robust, and stable to classify high dimensional and correlated data. These explanations provide a justification why Random Forest yielded better results in comparison with other classifiers [182].

Regarding this study, PCA was very efficient and produced better results than AE. In comparison with AE, the PCA approach is restricted to a linear mapping, whereas the AE can have a nonlinear encoder/decoder architecture.

As a future direction, this research will also serve as a base for further studies and investigations towards developing efficient IDS's from various intrusion detection datasets. Furthermore, the trained models could be extended to implement an IDS for online anomaly-based detection. From our experiments, we found that PCA is superior, faster, more interpretable and can reduce the dimensionality of the data to as low as 2 components.

The long training time and limited computational resources formed a barrier towards reducing the dimensionality beyond 59 features representation for the AE approach. Future research should therefore concentrate on the investigation of combating these barriers and to go below 59 features after reduction.

Further research might explore machine learning fault tolerance. Fault tolerance enables a system to continue operating properly in the event of failure or faults within any of its components. Fault tolerance can be achieved through several techniques. One aspect of fault tolerance in our system is the ability of the designed approach to detect a large set of well-known attacks. Our models have been trained to detect the 14 up-to-date and well-known type of attacks. Furthermore, fault tolerance can be achieved by adopting the majority voting technique [93, 218].

Moreover, an approach to quantify the resilience of machine learning classifiers was introduced in [224]. The association of these factors can be investigated as future studies.

7.1.3 CIDDS-001

A common problem that affects machine learning practically is the imbalanced class distribution problem as a result of disproportional classes. This dissertation was set out to design an efficient anomaly-based intrusion detection system on the imbalanced CIDDS-001 dataset and to compare the ways of treating imbalanced class distributions with the original class distributions. To compare solutions, we used alternative metrics such as Precision, Recall, Geometric Mean, Detection Rate, and the Combined metric along with Accuracy. The experiments confirm that DNN along with down-sampling and class-balancer lead to effective results in terms of accuracy of 99.65%. Moreover, Random Forest with less number of samples achieved accuracy of 99.99%. RF is more effective for real-time data fusion and applications for smaller sample sizes. It is noticed from this study that the class distribution has a light impact on the classification process. We believe that this is due to the nature of the dataset; which has a limited spectrum of attacks, binary classification, features sets, and finally the classifier type. However, more research on this topic needs to be

undertaken considering other class labels (Normal, Attacker, Victim, Suspicious and Unknown) that are embedded in CIDDS-001.

7.1.4 Performance Metrics

In this dissertation, the IDS's performance was evaluated through a set of metrics such as FAR, Acc., DR, F-M, G-Mean, Precision, Recall, and Kappa statistics.

With the imbalanced class distribution problem, the overall Accuracy metric may fail to provide adequate information about the performance of the classifier. Furthermore, the Accuracy is very sensitive to the class distribution and might be misleading in some way. Hence, our dissertation introduced the $Combined_{Mc}$ metric. The extensive experimental results show that the $Combined_{Mc}$ is able to give better insight on the evaluation of variant systems for selecting the best among them.

As a future research direction, there is a need to develop new metrics that must consider memory usage, computational power, detection latency, and power consumption in evaluating the IDS performance. These metrics are important for future implementations of IDS in both software-based machine learning and hardware-based behavior rules co-designs.

7.1.5 Imbalanced Class Problems

With respect to the imbalanced class distribution problem, this dissertation examined the imbalanced class distribution's problems through using three datasets that reflect realistic imbalanced classes. A variety of approaches to alleviate the imbalanced classes were examined including re-sample with replacement, re-sample without replacement, class balancing, over-sample the minority class by creating new synthetic instances, and One Vs All. The dissertation tested these approaches utilizing the single classifier model, ensemble classifier model, voting, and stacking. The

dissertation also introduced a uniform distribution based balancing approach to mitigate the problems of imbalanced classes.

Future studies on the current topic that investigate cost sensitive learning and important sampling approaches are recommended. Furthermore, more effort is needed to further evaluate the number of instances and class distribution in the testing set when learning from the imbalanced classes.

As network traffic IDS datasets are considered big data, deep learning approaches for intrusion detection seem to be the future trend as viable solutions. In the future, deep learning approaches could also be investigated further for anomaly based intrusion detection. Software-defined-networks (SDN) are increasingly growing becoming the future of networks. Deep learning approaches can be designed in SDN's for IDS.

7.2 Intrusion Detection: Software Machine Learning and Hardware Rules Based Co-Designs in MCPS IDS

The medical CPS IDS case study determined the effect of using a hardware approach to detect malicious behavior in sensors and actuators that are embedded in medical devices. The experimental results confirmed that the specification behavior rules can be utilized to build a hardware monitoring tool that can identify the expected normal behavior of a device and detect any deviation from its normal behavior. Furthermore, we showed through our analysis that our model is consistent with two dominant design requirements for next-generation high-end applications; lower power consumption and higher bandwidth. The reconfigurable nature of FPGA allows to modify and update the whole design according to the set of behavior rules, which outperform software-based approaches that are difficult to update. In addition, hardware approaches are difficult to be hacked. One of the most significant findings to

emerge from this study is that a hardware-based specification rules approach can be used to identify malicious behavior.

The hardware-based behavior specification rules approach yielded better results compared to the software-based machine learning approach for the same MCPS in detecting malicious behaviors. It also performs in near real-time. However, if new rules govern the CPS, the state machine for the behavior rules specifications monitoring tool should be redesigned, and the FPGA reconfigured and reprogrammed. A recommendation is that a hybrid approach be used where the rule-based approach (possibly embedded hardware) is host-based and the machine-learning approach is used at the network level (near the routers).

REFERENCES

[1] M. N. Chowdhury and K. Ferens, "A computational approach for detecting intrusion in communication network using machine learning," in *International Conference on Advances on Applied Cognitive Computing ACC'17*, 2017.

[2] Kolias, Constantinos and Kambourakis, Georgios and Stavrou, Angelos and Gritzalis, Stefanos, "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2016.

[3] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), 2015*, pp. 1–6, IEEE.

[4] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[5] N. Moustafa and J. Slay, "The significant features of the unsw-nb15 and the kdd99 data sets for network intrusion detection systems," in *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2015 4th International Workshop on*, pp. 25–31, IEEE.

[6] D. W. Vilela, T. F. Ed'Wilson, A. A. Shinoda, N. V. de Souza Araujo, R. de Oliveira, and V. E. Nascimento, "A dataset for evaluating intrusion detection systems in ieee 802.11 wireless networks," in *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pp. 1–5, Ieee.

[7] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Flow-based benchmark data sets for intrusion detection," in *Proceedings of the 16th European Conference on Cyber Warfare and Security*, pp. 361–369, 2017.

[8] A. H. L. Iman Sharafaldin and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *(ICISSP2018)*, January 2018.

[9] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "Ugr'16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, vol. 73, pp. 411–424, 2018.

[10] L. A. Jeni, J. F. Cohn, and F. De La Torre, "Facing imbalanced data–recommendations for the use of performance metrics," in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pp. 245–251, IEEE, 2013.

[11] F. Guillet and H. J. Hamilton, *Quality measures in data mining*, vol. 43. Springer, 2007.

[12] Y. Liu, J. Cheng, C. Yan, X. Wu, and F. Chen, "Research on the matthews correlation coefficients metrics of personalized recommendation algorithm evaluation," *International Journal of Hybrid Information Technology*, vol. 8, no. 1, pp. 163–172, 2015.

[13] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, ACM, 2006.

[14] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[15] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes," *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, 2011.

[16] A. Ben-David, "About the relationship between roc curves and cohen's kappa," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 6, pp. 874–882, 2008.

[17] W. Zong, G.-B. Huang, and Y. Chen, "Weighted extreme learning machine for imbalance learning," *Neurocomputing*, vol. 101, pp. 229–242, 2013.

[18] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

[19] R. Espíndola and N. Ebecken, "On extending f-measure and g-mean metrics to multi-class problems," *WIT Transactions on Information and Communication Technologies*, vol. 35, 2005.

[20] T. Hamed, R. Dara, and S. C. Kremer, "Network intrusion detection system based on recursive feature addition and bigram technique," *Computers Security*, vol. 73, pp. 137–155, 2018.

[21] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[22] R. Mitchell and R. Chen, "Behavior-rule based intrusion detection systems for safety critical smart grid applications," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1254–1263, 2013.

[23] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 2, pp. 577–583, 2008.

[24] W. Hu, J. Gao, Y. Wang, O. Wu, and S. Maybank, "Online adaboost-based parameterized methods for dynamic distributed network intrusion detection," *IEEE Transactions on Cybernetics*, vol. 44, no. 1, pp. 66–82, 2014.

[25] M. A. Hall, *Correlation-based feature selection for machine learning*. PhD thesis, University of Waikato Hamilton, 1999.

[26] G. Kaur and A. Chhabra, "Improved j48 classification algorithm for the prediction of diabetes," *International Journal of Computer Applications*, vol. 98, no. 22, 2014.

[27] S. Sahu and B. M. Mehtre, "Network intrusion detection system using j48 decision tree," in *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pp. 2023–2026, IEEE, 2015.

[28] J. Friedman, T. Hastie, R. Tibshirani, *et* al., "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[29] M. H. Kamarudin, C. Maple, T. Watson, and N. S. Safa, "A logitboost-based algorithm for detecting known and unknown web attacks," *IEEE Access*, vol. 5, pp. 26190–26200, 2017.

[30] M. Belouch, S. El Hadaj, and M. Idhammad, "A two-stage classifier approach using reptree algorithm for network intrusion detection," *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS*, vol. 8, no. 6, pp. 389–394, 2017.

[31] A. Boulaiche and K. Adi, "An auto-learning approach for network intrusion detection," *Telecommunication Systems*, pp. 1–18, 2017.

[32] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[33] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report, James P. Anderson Company*, 1980.

[34] M. E. Whitman and H. J. Mattord, *Principles of information security*. Cengage Learning, 2011.

[35] R. R. Mitchell III, *Design and Analysis of Intrusion Detection Protocols in Cyber Physical Systems*. PhD thesis, Virginia Tech, 2013.

[36] R. Abdulhammed, M. Faezipour, and K. Elleithy, *Intrusion Detection in Self-Organizing Networks: A Survey*, ch. 13, pp. 393–449. NewYork: CRC Press Taylor Francis Group, 2017.

[37] A.-S. K. Pathan, *Security of self-organizing networks: MANET, WSN, WMN, VANET*. CRC press, 2016.

[38] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[39] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516–524, 2010.

[40] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[41] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[42] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *Ieee communications surveys & tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[43] R. Abdulhammed, M. Faezipour, and K. M. Elleithy, "Network intrusion detection using hardware techniques: A review," in *Systems, Applications and Technology Conference (LISAT), 2016 IEEE Long Island*, pp. 1–7, IEEE, 2016.

[44] K. Hwang, M. Cai, Y. Chen, and M. Qin, "Hybrid intrusion detection with weighted signature generation over anomalous internet episodes," *IEEE Transactions on dependable and secure computing*, vol. 4, no. 1, pp. 41–55, 2007.

[45] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1690–1700, 2014.

[46] P. Uppuluri and R. Sekar, "Experiences with specification-based intrusion detection," Recent Advances in Intrusion Detection, pp. 172–189, Springer Berlin Heidelberg, 2001.

[47] A. P. R. da Silva, M. H. Martins, B. P. Rocha, A. A. Loureiro, L. B. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Proceedings of the 1st ACM international workshop on Quality of service security in wireless and mobile networks*, pp. 16–23, ACM, 2005.

[48] J. O. Nehinbe, "A critical evaluation of datasets for investigating idss and ipss researches," in *Cybernetic Intelligent Systems (CIS), 2011 IEEE 10th International Conference on*, pp. 92–97, IEEE.

[49] M. Najafabadi, T. Khoshgoftaar, and C. Kemp, "The importance of representative network data on classification models for the detection of specific network attacks," in *21st ISSAT Int. Conf. Reliability and Quality in Design*, pp. 59–64, 2015.

[50] M. E. Aminanto and K. Kim, "Improving detection of wi-fi impersonation by fully unsupervised deep learning," in *Information Security Applications: 18th International Workshop, WISA 2017*, 2017.

[51] R. Zuech, T. M. Khoshgoftaar, N. Seliya, M. M. Najafabadi, and C. Kemp, "A new intrusion detection benchmarking system," in *FLAIRS Conference*, pp. 252–256, 2015.

[52] C. Wheelus, T. M. Khoshgoftaar, R. Zuech, and M. M. Najafabadi, "A session based approach for aggregating network traffic data–the santa dataset," in *2014 IEEE International Conference on Bioinformatics and Bioengineering*, pp. 369–378, IEEE, 2014.

[53] K. Cup, "Intrusion detection data set," *The UCI KDD Archive Information and Computer Science University of California, Irvine. DOI= http://kdd. ics. uci. edu/databases/kddcup99*, 1999.

[54] K. Cup, "Data/the uci kdd archive, information and computer science," *University of California, Irvine*, 1999.

[55] S. D. Bay, D. Kibler, M. J. Pazzani, and P. Smyth, "The uci kdd archive of large data sets for data mining research and experimentation," *ACM SIGKDD explorations newsletter*, vol. 2, no. 2, pp. 81–85, 2000.

[56] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pp. 1–6, IEEE.

[57] H. Gharaee and H. Hosseinvand, "A new feature selection ids based on genetic algorithm and svm," in *Telecommunications (IST), 2016 8th International Symposium on*, pp. 139–144, IEEE.

[58] S. Guha, S. S. Yau, and A. B. Buduru, "Attack detection in cloud infrastructures using artificial neural network with genetic feature selection," in *Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C*, pp. 414–419, IEEE, 2016.

[59] M. M. Aravind and V. Kalaiselvi, "Design of an intrusion detection system based on distance feature using ensemble classifier," in *Signal Processing, Communication and Networking (ICSCN), 2017 Fourth International Conference on*, pp. 1–6, IEEE, 2017.

[60] M. Idhammad, K. Afdel, and M. Belouch, "Dos detection method based on artificial neural networks," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 4, pp. 465–471, 2017.

[61] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on*, pp. 97–103, IEEE.

[62] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Transactions on Big Data*, 2017.

[63] N. Moustafa and J. Slay, "A hybrid feature selection for network intrusion detection systems: Central points," *arXiv preprint arXiv:1707.05505*, 2017.

[64] S. M. H. Bamakan, H. Wang, and Y. Shi, "Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem," *Knowledge-Based Systems*, vol. 126, pp. 113–126, 2017.

[65] N. Moustafa, G. Creech, and J. Slay, *Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models*, pp. 127–156. Springer, 2017.

[66] T. Janarthanan and S. Zargari, "Feature selection in unsw-nb15 and kddcup'99 datasets," in *Industrial Electronics (ISIE), 2017 IEEE 26th International Symposium on*, pp. 1881–1886, IEEE.

[67] N. Moustafa, G. Creech, and J. Slay, "Anomaly detection system using beta mixture models and outlier detection," in *Progress in Computing, Analytics and Networking*, pp. 125–135, Springer, 2018.

[68] K. Mwitondi and S. Zargari, "A repeated sampling and clustering method for intrusion detection," in *International Conference in Data Mining (DMIN'17)*, pp. 91–96, CSREA Press, 2017.

[69] V. Timčenko and S. Gajin, "Ensemble classifiers for supervised anomaly based network intrusion detection," in *Intelligent Computer Communication and Processing (ICCP), 2017 13th IEEE International Conference on*, pp. 13–19, IEEE.

[70] K. Nahiyan, S. Kaiser, K. Ferens, and R. McLeod, "A multi-agent based cognitive approach to unsupervised feature extraction and classification for network intrusion detection," in *Int'l Conf. on Advances on Applied Cognitive Computing*, pp. 25–30, CSREA Press, 2017.

[71] M. Al-Zewairi, S. Almajali, and A. Awajan, "Experimental evaluation of a multilayer feed-forward artificial neural network classifier for network intrusion detection system," in *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pp. 167–172, IEEE, 2017.

[72] A. Agrawal, S. Mohammed, and J. Fiaidhi, "Developing data mining techniques for intruder detection in network traffic," *International Journal of Security and Its Applications*, vol. 10, no. 8, pp. 335–342, 2016.

[73] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques," *arXiv preprint arXiv:1711.02825*, 2017.

[74] I. Benmessahel, K. Xie, and M. Chellal, "A new evolutionary neural networks based on intrusion detection systems using multiverse optimization," *Applied Intelligence*, pp. 1–13, 2017.

[75] A. Valero León, "Insides: A new machine learning-based intrusion detection system," 2017.

[76] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz, "Cyberattack detection in mobile cloud computing: A deep learning approach," *arXiv preprint arXiv:1712.05914*, 2017.

[77] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," *Neural Computing and Applications*, pp. 1–11, 2017.

[78] K. Anusha and E. Sathiyamoorthy, "Omamids: ontology based multi-agent model intrusion detection system for detecting web service attacks," *Journal of Applied Security Research*, vol. 11, no. 4, pp. 489–508, 2016.

[79] L. Van Efferen and A. M. Ali-Eldin, "A multi-layer perceptron approach for flow-based anomaly detection," in *Networks, Computers and Communications (ISNCC), 2017 International Symposium on*, pp. 1–6, IEEE.

[80] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," *Future Generation Computer Systems*, vol. 79, pp. 558–574, 2018.

[81] V. Varadharajan, U. Tupakula, and K. Karmakar, "Secure monitoring of patients with wandering behaviour in hospital environments," *IEEE Access*, 2017.

[82] C. Khammassi and S. Krichen, "A ga-lr wrapper approach for feature selection in network intrusion detection," *Computers & Security*, vol. 70, pp. 255–277, 2017.

[83] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," in *Information Science and Security (ICISS), 2016 International Conference on*, pp. 1–5, IEEE, 2016.

[84] P. Mishra, E. S. Pilli, V. Varadharajant, and U. Tupakula, "Nvcloudids: A security architecture to detect intrusions at network and virtualization layer in cloud environment," in *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*, pp. 56–62, IEEE, 2016.

[85] M. M. Aravind and V. Kalaiselvi, "Design of an intrusion detection system based on distance feature using ensemble classifier," in *Signal Processing, Communication and Networking (ICSCN), 2017 Fourth International Conference on*, pp. 1–6, IEEE, 2017.

[86] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on*, pp. 97–103, IEEE.

[87] S. Siddiqui, "Cognitive artificial intelligence–a complexity based machine learning approach for advanced cyber threats," 2016.

[88] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for wi-fi impersonation detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 621–636, 2018.

[89] U. S. K. P. M. Thanthrige, J. Samarabandu, and X. Wang, "Machine learning techniques for intrusion detection on public dataset," in *Electrical and Computer Engineering (CCECE), 2016 IEEE Canadian Conference on*, pp. 1–4, IEEE.

[90] A. H. Lashkari, M. M. S. Danesh, and B. Samadi, "A survey on wireless security protocols (wep, wpa and wpa2/802.11 i)," in *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pp. 48–52, IEEE, 2009.

[91] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. Alessa, "Effective features selection and machine learning classifiers for improved wireless intrusion detection," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, IEEE, 2018.

[92] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. AbuMallouh, "Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic," *IEEE Sensors Letters*, vol. 3, no. 1, pp. 1–4, 2019.

[93] G. James, *Majority vote classifiers: theory and applications*. PhD thesis, Stanford University, 1998.

[94] M. E. Aminanto and K. Kim, "Detecting impersonation attack in wifi networks using deep learning approach," in *International Workshop on Information Security Applications*, pp. 136–147, Springer, 2016.

[95] U. S. K. P. M. Thanthrige, "Hidden markov model based intrusion alert prediction," Master's thesis, University of Western Ontario, 2016.

[96] C. Kolias, V. Kolias, and G. Kambourakis, "Termid: a distributed swarm intelligence-based approach for wireless intrusion detection," *International Journal of Information Security*, vol. 16, no. 4, pp. 401–416, 2017.

[97] M. E. Aminanto and K. Kim, "Detecting active attacks in wifi network by semi-supervised deep learning," in *Conference on Information Security and Cryptography 2017 Winter*, 2016.

[98] M. E. Aminanto, H. Tanuwidjaja, P. D. Yoo, and K. Kim, "Weighted feature selection techniques for detecting impersonation attack in wi-fi networks," in *Proc. Symp. Cryptogr. Inf. Secur.(SCIS)*, pp. 1–8, 2017.

[99] M. Usha and P. Kavitha, "Anomaly based intrusion detection for 802.11 networks with optimal features using svm classifier," *Wireless Networks*, vol. 23, no. 8, pp. 2431–2446, 2017.

[100] B. A. Tama and K.-H. Rhee, "classifier ensemble design with rotation forest to enhance attack detection of ids in wireless network," in *Information Security (AsiaJCIS), 2016 11th Asia Joint Conference on*, pp. 87–91, IEEE.

[101] B. A. Tama and K.-H. Rhee, "A detailed analysis of classifier ensembles for intrusion detection in wireless network," *Journal of Information Processing Systems*, vol. 13, no. 5, 2017.

[102] B. A. Tama and K.-H. Rhee, *A Novel Anomaly Detection Method in Wireless Network Using Multi-level Classifier Ensembles*, pp. 452–458. Springer, 2017.

[103] A. Verma and V. Ranga, "Statistical analysis of cidds-001 dataset for network intrusion detection systems using distance-based machine learning," *Procedia Computer Science*, vol. 125, pp. 709–716, 2018.

[104] R. Theron, R. Magán-Carrión, J. Camacho, and G. M. Fernández, "Network-wide intrusion detection supported by multivariate analysis and interactive visualization," in *Visualization for Cyber Security (VizSec), 2017 IEEE Symposium on*, pp. 1–8, IEEE, 2017.

[105] D. Kaleem and K. Ferens, "A cognitive multi-agent model to detect malicious threats," in *International Conference on Advances Applied Cognitive Computing ,2016*, pp. 58–63, CSREA, 2016.

[106] V. L. Thing, "Ieee 802.11 network anomaly detection and attack classification: A deep learning approach," in *Wireless Communications and Networking Conference (WCNC), 2017 IEEE*, pp. 1–6, IEEE.

[107] "Canadian Institute of Cybersecurity, University of New Brunswick, Available online: https://www.unb.ca/cic/research/applications.html#CICFlowMeter (accessed on 23 January 2019)," 2017.

[108] CIC, "Canadian Institute of Cybersecurity. List of Extracted Traffic Features by CICFlowMeter-V3. 2017, Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 23 January 2019)," 2017.

[109] R. Vijayan and, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Computers & Security*, vol. 77, pp. 304–314, 2018.

[110] B. J. Radford and B. D. Richardson, "Sequence aggregation rules for anomaly detection in computer network traffic," *arXiv preprint arXiv:1805.03735*, 2018.

[111] D. Lavrova, P. Semyanov, A. Shtyrkina, and P. Zegzhda, "Wavelet-analysis of network traffic time-series for detection of attacks on digital production infrastructure," in *SHS Web of Conferences*, vol. 44, p. 00052, EDP Sciences, 2018.

[112] G. Watson, "A comparison of header and deep packet features when detecting network intrusions," tech. rep., 2018.

[113] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," in *International Symposium on Computer and Information Sciences*, pp. 141–149, Springer, 2018.

[114] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018.

[115] A. Spark, "PySpark 2.4.0 documentation Available online: https://spark.apache.org/docs/latest/api/python/index.html# (accessed on 10 November 2018)," 2018.

[116] A. Bansal, *DDR Scheme and LSTM RNN Algorithm for Building an Efficient IDS*. PhD thesis, 2018.

[117] T. Chen, T. He, M. Benesty, *et* al., "Xgboost: extreme gradient boosting," *R package version 0.4-2*, pp. 1–4, 2015.

[118] T. Hothorn, K. Hornik, and A. Zeileis, "ctree: Conditional inference trees," *The Comprehensive R Archive Network*, 2015.

[119] A. Bansal and S. Kaur, "Extreme gradient boosting based tuning for classification in intrusion detection systems," in *International Conference on Advances in Computing and Data Sciences*, pp. 372–380, Springer, 2018.

[120] D. Aksu and M. A. Aydin, "Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pp. 77–80, IEEE, 2018.

[121] S. Ustebay, Z. Turgut, and M. A. Aydin, "Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, pp. 71–76, IEEE, 2018.

[122] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Creation of flow-based data sets for intrusion detection," *Journal of Information Warfare*, vol. 16, pp. 40–53, 2017.

[123] B. A. Tama and K.-H. Rhee, "Attack classification analysis of iot network via deep learning approach," *Research Briefs on Information & Communication Technology Evolution (ReBICTE)*, 2018.

[124] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on*, pp. 114–120, IEEE, 2017.

[125] M. Idhammad, K. Afdel, and M. Belouch, "Detection system of http ddos attacks in a cloud environment based on information theoretic entropy and random forest," *Security and Communication Networks*, vol. 2018, 2018.

[126] M. Idhammad and M. Afdel, Karim Belouch, "Distributed intrusion detection system for cloud environments based on data mining techniques," *Procedia Computer Science*, vol. 127, pp. 35–41, 2018.

[127] L. Nicholas, S. Y. Ooi, Y. H. Pang, S. O. Hwang, and S.-Y. Tan, "Study of long short-term memory in flow-based network intrusion detection system," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–11, 2018.

[128] A. Verma and V. Ranga, "On evaluation of network intrusion detection systems: Statistical analysis of cidds-001 dataset using machine learning techniques.," *Pertanika Journal of Science & Technology*, vol. 26, no. 3, 2018.

[129] S. Rodda and U. S. R. Erothi, "Class imbalance problem in the network intrusion detection systems," in *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on*, pp. 2685–2688, IEEE, 2016.

[130] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "Nsl-kdd dataset," *Available on http://www. unb. ca/research/iscx/dataset/iscx-NSL-KDD-dataset. html),[Accessed on 28 Feb. 2016]*, 2012.

[131] R. Balasubramanian and S. Joseph, "Intrusion detection on highly imbalance big data using tree based real time intrusion detection system: effects and solutions," *Int. J. Adv. Res. Comput. Commun. Eng*, vol. 5, no. 2, pp. 27–32, 2016.

[132] D. A. Cieslak, N. V. Chawla, and A. Striegel, "Combating imbalance in network intrusion datasets.," in *GrC*, pp. 732–737, 2006.

[133] D. Adamu Teshome and V. S. Rao, "A cost sensitive machine learning approach for intrusion detection," *Global Journal of Computer Science and Technology*, 2014.

[134] V. Engen, *Machine learning for network based intrusion detection: an investigation into discrepancies in findings with the KDD cup'99 data set and multi-objective evolution of neural network classifier ensembles from imbalanced data.* PhD thesis, Bournemouth University, 2010.

[135] M. R. Parsaei, S. M. Rostami, and R. Javidan, "A hybrid data mining approach for intrusion detection on imbalanced nsl-kdd dataset," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 6, pp. 20–25, 2016.

[136] K.-C. Khor, C.-Y. Ting, and S. Phon-Amnuaisuk, "The effectiveness of sampling methods for the imbalanced network intrusion detection data set," in *Recent Advances on Soft Computing and Data Mining*, pp. 613–622, Springer, 2014.

[137] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, *e*t al., "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[138] B. Yan, G. Han, M. Sun, and S. Ye, "A novel region adaptive smote algorithm for intrusion detection on imbalanced problem," in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1281–1286, IEEE, 2017.

[139] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference*, pp. 731–736, IEEE, 2010.

[140] R. Abdulhammed, M. Faezipour, and K. Elleithy, "Malicious behavior monitoring of embedded medical devices," in *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–6, 2017.

[141] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46, pp. 1–12, Citeseer.

[142] P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling response to anomalous live disturbances," in *Proceedings of the 20th national information systems security conference*, pp. 353–365, 1997.

[143] M. Roesch *et* al., "Snort: Lightweight intrusion detection for networks.," in *Lisa*, vol. 99, pp. 229–238, 1999.

[144] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta, "A multidimensional critical state analysis for detecting intrusions in scada systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 179–186, 2011.

[145] R. Mitchell and I.-R. Chen, "Specification based intrusion detection for unmanned aircraft systems," in *Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications*, pp. 31–36, ACM, 2012.

[146] H. Bao, R. Lu, B. Li, and R. Deng, "Blithe: Behavior rule-based insider threat detection for smart grid," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 190–205, 2016.

[147] B. Asfaw, D. Bekele, B. Eshete, A. Villafiorita, and K. Weldemariam, "Host-based anomaly detection for pervasive medical systems," in *Risks and Security of Internet and Systems (CRiSIS), 2010 Fifth International Conference on*, pp. 1–8, IEEE, 2010.

[148] W.-S. Yang and S.-Y. Hwang, "A process-mining framework for the detection of healthcare fraud and abuse," *Expert Systems with Applications*, vol. 31, no. 1, pp. 56–68, 2006.

[149] R. Mitchell and I. R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 16–30, 2015.

[150] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *expert systems with applications*, vol. 36, no. 10, pp. 11994–12000, 2009.

[151] J. Martınez, O. Fuentes, and L. E. Erro, "Using c4. 5 as variable selection criterion in classification tasks," in *Proceedings of the 9th IASTED International Conference on Artificial Intelligence and Soft Computing*, pp. 171–176, 2005.

[152] R. Diao and Q. Shen, "Feature selection with harmony search," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 6, pp. 1509–1523, 2012.

[153] K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, C. R. Pereira, J. P. Papa, and A. Xavier FalcÃ£o, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Information Sciences*, vol. 294, pp. 95–108, 2015.

[154] Z. W. Geem, *Music-inspired harmony search algorithm: theory and applications*, vol. 191. Springer, 2009.

[155] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M. N. Bilbao, S. Salcedo-Sanz, and Z. W. Geem, "A survey on applications of the harmony search algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1818–1831, 2013.

[156] H.-H. Gao, H.-H. Yang, and X.-Y. Wang, "Ant colony optimization based network intrusion feature selection and detection," in *Machine Learning and Cyber-*

netics, 2005. Proceedings of 2005 International Conference on, vol. 6, pp. 3871–3875, IEEE.

[157] H. R. Kanan and K. Faez, "An improved feature selection method based on ant colony optimization (aco) evaluated on face recognition system," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 716–725, 2008.

[158] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[159] S. Palanisamy and S. Kanmani, "Artificial bee colony approach for optimizing feature selection," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 3, p. 432, 2012.

[160] L. Xu, P. Yan, and T. Chang, "Best first strategy for feature selection," in *Pattern Recognition, 1988., 9th International Conference on*, pp. 706–708, IEEE.

[161] M. A. Hall and L. A. Smith, "Feature subset selection: a correlation based filter approach," 1997.

[162] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *arXiv preprint arXiv:1403.2877*, 2014.

[163] I. K. Fodor, "A survey of dimension reduction techniques," *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory*, vol. 9, pp. 1–18, 2002.

[164] D. Xia, S. Yang, and C. Li, "Intrusion detection system based on principal component analysis and grey neural networks," in *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 2, pp. 142–145, April 2010.

[165] R. Rosaria, I. Adae, A. Hart, and M. Berthold, "Seven techniques for dimensionality reduction," 2014.

[166] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative review," *J Mach Learn Res*, vol. 10, pp. 66–71, 2009.

[167] P. Bertens, "Rank ordered autoencoders," *arXiv preprint arXiv:1605.01749*, 2016.

[168] A. M. Martínez and A. C. Kak, "Pca versus lda," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 2, pp. 228–233, 2001.

[169] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[170] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

[171] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, p. 4, ACM, 2014.

[172] A. Makhzani, *Unsupervised representation learning with autoencoders*. PhD thesis, 2018.

[173] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[174] I. M. Johnstone and A. Y. Lu, "Sparse principal components analysis," *Unpublished manuscript*, vol. 7, 2004.

[175] K. K. Vasan and B. Surendiran, "Dimensionality reduction using principal component analysis for network intrusion detection," *Perspectives in Science*, vol. 8, pp. 510–512, 2016.

[176] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[177] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[178] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[179] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[180] F. Tang and H. Ishwaran, "Random forest missing data algorithms," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 10, no. 6, pp. 363–377, 2017.

[181] L. Breiman and A. Cutler, "Random forest-manual," *Online: http://www. stat. berkeley. edu/˜ breiman/RandomForests/cc_ manual. htm*, 2004.

[182] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016.

[183] B. Zenko, L. jupco Todorovski, and S. Dzeroski, "A comparison of stacking with meta decision trees to bagging, boosting, and stacking with other methods," in *ICDM*, p. 669, IEEE, 2001.

[184] A. J. Ferreira and M. A. Figueiredo, *Boosting algorithms: A review of methods, theory, and applications*, pp. 35–85. Springer, 2012.

[185] S. Menard, *Applied logistic regression analysis*, vol. 106. SAGE publications, 2018.

[186] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Machine learning*, vol. 11, no. 1, pp. 63–90, 1993.

[187] S. Sayad, *Real time data mining*. Self-Help Publishers Cambridge, 2011.

[188] R. Kohavi and J. R. Quinlan, "Data mining tasks and methods: Classification: decision-tree discovery," in *Handbook of data mining and knowledge discovery*, pp. 267–276, Oxford University Press, Inc., 2002.

[189] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[190] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[191] V. García, J. S. Sánchez, and R. A. Mollineda, "On the effectiveness of preprocessing methods when dealing with different levels of class imbalance," *Knowledge-Based Systems*, vol. 25, no. 1, pp. 13–21, 2012.

[192] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, *e*t al., "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.

[193] S.-J. Yen and Y.-S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," in *Intelligent Control and Automation*, pp. 731–740, Springer, 2006.

[194] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[195] P. Bermejo, J. A. Gámez, and J. M. Puerta, "Improving the performance of naive bayes multinomial in e-mail foldering by introducing distribution-based balance of datasets," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2072–2080, 2011.

[196] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3: 322, pp. 1–27, 2019.

[197] D. M. Zuckerman, P. Brown, and S. E. Nissen, "Medical device recalls and the fda approval process," *Archives of internal medicine*, vol. 171, no. 11, pp. 1006–1011, 2011.

[198] R. Mitchell and R. Chen, "Behavior rule based intrusion detection for supporting secure medical cyber physical systems," in *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pp. 1–7, IEEE.

[199] N. Ramesh, F. Mesadi, M. Cetin, and T. Tasdizen, "Disjunctive normal shape models," in *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, pp. 1535–1539, IEEE.

[200] R. Abdulhammed, M. Faezipour, A. Abuzneid, and A. Alessa, "Enhancing wireless intrusion detection using machine learning classification with reduced attribute sets," in *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 524–529, IEEE, 2018.

[201] N. P. Neelakantan, C. Nagesh, and M. Tech, "Role of feature selection in intrusion detection systems for 802.11 networks," *International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN)*, vol. 1, no. 1, pp. 98–101, 2011.

[202] M. E. Aminanto, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Wi-fi intrusion detection using weighted-feature selection for neural networks classifier," pp. 99–104, 2017.

[203] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 155–164, ACM, 1999.

[204] J. Zhu, Y. Ming, Y. Song, and S. Wang, "Mechanism of situation element acquisition based on deep auto-encoder network in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 13, no. 3, p. 1550147717699625, 2017.

[205] M. Al-Qatf, M. Alhabib, K. Al-Sabahi, *e*t al., "Deep learning approach combining sparse autoen-coder with svm for network intrusion detection," *IEEE Access*, 2018.

[206] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[207] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, and J. Ma, "Su-ids: A semi-supervised and unsupervised framework for network intrusion detection," in *International Conference on Cloud Computing and Security*, pp. 322–334, Springer, 2018.

[208] R. Abdulhammed, M. Faezipour, H. Musafer, and A. Abuzneid, "Efficient network intrusion detection using pca-based dimensionality reduction of features,"

in *2019 International Symposium on Networks, Computers and Communications (ISNCC), accepted (to appear)*, IEEE, 2019.

[209] K. K. Vasan and B. Surendiran, "Dimensionality reduction using principal component analysis for network intrusion detection," *Perspectives in Science*, vol. 8, pp. 510–512, 2016.

[210] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook*, pp. 875–886, Springer, 2009.

[211] N. Japkowicz *et* al., "Learning from imbalanced data sets: a comparison of various strategies," in *AAAI workshop on learning from imbalanced data sets*, vol. 68, pp. 10–15, Menlo Park, CA, 2000.

[212] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, p. 160035, 2016.

[213] D. Liu, M. Görges, and S. A. Jenkins, "University of queensland vital signs dataset: Development of an accessible repository of anesthesia patient monitoring data for research," *Anesthesia & Analgesia*, vol. 114, no. 3, pp. 584–589, 2012.

[214] V. Raghavan, P. Bollmann, and G. S. Jung, "A critical investigation of recall and precision as measures of retrieval system performance," *ACM Transactions on Information Systems (TOIS)*, vol. 7, no. 3, pp. 205–229, 1989.

[215] M. Buckland and F. Gey, "The relationship between recall and precision," *Journal of the American society for information science*, vol. 45, no. 1, pp. 12–19, 1994.

[216] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, 2007.

[217] Y. Qian, Y. Liang, M. Li, G. Feng, and X. Shi, "A resampling ensemble algorithm for classification of imbalance problems," *Neurocomputing*, vol. 143, pp. 57–67, 2014.

[218] B. Alotaibi and K. Elleithy, "A majority voting technique for wireless intrusion detection systems," in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1–6, IEEE, 2016.

[219] G. Louppe, *Understanding random forests: From theory to practice*. PhD thesis, 2014.

[220] J. Cohen, *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[221] K. R. Murphy, B. Myors, and A. Wolach, *Statistical power analysis: A simple and general model for traditional and modern hypothesis tests*. Routledge, 2014.

[222] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[223] J. A. Lozano, G. Santafé, and I. Inza, "Classifier performance evaluation and comparison," in *International Conference on Machine Learning and Applications (ICMLA 2010)*, 2010.

[224] E. Viegas, A. Santin, N. Neves, A. Bessani, and V. Abreu, "A resilient stream learning intrusion detection mechanism for real-time analysis of network traffic," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, IEEE, 2017.